

واستخداماته
في المعلوماتية الحيوية

٢٠ مشروع تعلم عميق في المعلوماتية الحيوية تم حلها وشرحها باستخدام بايثون

ترجمة واعداد: د. علاء طعيمة



بسمه تعالى

التعلم العميق واستخداماته فيِ المعلوماتية الحيوية

20 مشروع تعلم عميق في علم الجينوم وعلم البروتين تم حلها وشرحها باستخدام بايثون

ترجمة واعداد:

د. علاء طعيمة

مقدمة المؤلف

أظهر التعلم العميق نتائج واعدةفي مجال المعلوماتية الحياتية بمافي ذلك علم الجينوم والبروتينات. ومع ذلك، هناك نقصفي القوى العاملة الماهرةفي التعلم العميقفي هذا التخصص. سيساعد هذا الكتاب الباحثين وعلماء البيانات على التميز عن الآخرين وحل مشاكل العالم الحقيقيفي مجال المعلوماتية الحياتية من خلال مجموعة من المشاريع. يغطي الكتاب اهم نماذج التعلم العميق المهمة التي يشيع استخدامها من قبل مجتمع البحث ويخوضفي تفاصيل ماهيتها وكيفية عملها وتطبيقاتها العمليةفي علم الجينوم وعلم البروتينات.

سيكون هذه الكتاب مفيداً للغاية لأولئك الذين يعملونفي مجال الذكاء الاصطناعي الذين يحاولون شق طريقهمفي مجال المعلوماتية الحياتية ولأولئك الذين يعملونفي مجال المعلوماتية الحياتية الذين يحاولون شق طريقهمفي مجال الذكاء الاصطناعي.

في هذا الكتاب، تنقل مشاريع التعلم العميقفي المعلوماتية الحيوية كل المعرفة اللازمة لتنفيذ مشاريع التعلم العميقفي مختلف مجالات المعلوماتية الحيوية. كل مشروع من هذه المشاريع فريد من نوعه، مما يساعدك على إتقان الموضوع تدريجيًا. ستتعلم كيفية تصنيف تسلسلات الجينوم والبروتين باستخدام نماذج التعلم العميق وكذلك ستتعلم الكثير من المشاريع الجذابة الاخرى الي ستساعدكفي اكتساب المعرفة لتطوير المعلوماتية الحياتية باستخدام التعلم العميق.

لقد حاولت قدر المستطاع ان اترجم المشاريع الأكثر طرحاًفي مجال التعلم العميق للمعلومات الحيوية مع الشرح المناسب والكافي، ومع هذا يبقى عملاً بشرياً يحتمل النقص، فاذا كان لديك أي ملاحظات حول هذا الكتاب، فلا تتردد بمراسلتنا عبر بريدنا الالكتروني alaa.taima@qu.edu.iq .

نأمل ان يساعد هذا الكتاب كل من يريد ان يدخلفي مجالات التعلم العميق والمعلوماتية الحيوية ومساعدة القارئ العربي على تعلم هذا المجالات. اسأل الله التوفيقفي هذا العمل لأثراء المحتوى العربي الذي يفتقر أشد الافتقار إلى محتوى جيد ورصينفي مجال التعلم الآلي والتعلم العميق وعلم البيانات. ونرجو لك الاستمتاع مع الكتاب ولا تنسونا من صالح الدعاء.

د. علاء طعيمة

كلية علوم الحاسوب وتكنولوجيا المعلومات

جامعة القادسية

العراق

المحتويات

1) تطبيق خوارزميات التعلم الآلي لتصنيف بيانات الجينوم Apply Machine Learning Algorithms for Genomics Data Classification!

المقدمة

1. نظرة عامة

في مجالات الاحياء الجزيئية molecular biology وعلم الوراثة genetics، الجينوم genome هو كل المواد الجينية للكائن الحي. يتكون من حمض نووي ريبوزي منقوص الأكسجين DNA (أو حمض نووي ريبوزي RNAفي فيروسات RNA). يتضمن الجينوم كلاً من الجينات genes (مناطق الترميز coding regions) والحمض النووي غير المشفر non-coding DNA، بالإضافة إلى الحمض النووي للميتوكوندريا mitochondrial DNA والحمض النووي للبلاستيدات الخضراء chloroplast DNA. تسمى دراسة الجينوم علم الجينوم genomics. يتضمن تفاعلات الجينات مع بعضها البعض ومع بيئة الشخص. تُستخدم البيانات الجينوميةفي مجال المعلوماتية الحيوية Bioinformatics لجمع وتخزين ومعالجة جينومات الكائنات الحية. تتطلب معالجة البيانات الجينومية قدرًا كبيرًا من تخزين البيانات وأجهزة وبرامج عالية الأداء للتحليل الإحصائي.

التعلم الآلي (ML) Machine Learning هو تطبيق للذكاء الاصطناعي Artificial Intelligence (AI) يتيح التعلم التلقائي والتحسين من التجربة دون البرمجة الصريحة والمعرفة ببيئة التعلم. الهدف الأهم هو تطوير ونشر نظام كمبيوتر يمكنه التعلم تلقائيًا دون أي تدخل بشري. بحكم التعريف يجب أن تكون قادرة على تعديل إجراءاتها من النتائج السابقة.

أصبح التعلم الآلي أحد الأساليب الرئيسية للعديد من مهام أبحاث الجينوم اليوم، بمافي ذلك:

1. وصف وتفسير مجموعات البيانات الجينومية واسعة النطاق.
2. شرح لمجموعة واسعة من عناصر تسلسل الجينوميات.
3. توقع تأثير الاختلاف الجيني على تسلسل DNA / RNA.
4. تحديد احتمالية الإصابة بمرض معين وكذلك تحديد الوراثة الجينية.
5. التعرف على الأنماط ووضع التنبؤات ووضع نموذج لتطور مرض معين أو علاجه.

يمكن أن تكون التطبيقات المستقبلية للتعلم الآليفي علم الجينوم: علم الصيدلة الجيني Pharmacogenomics، وأدوات الفحص الجيني لحديثي الولادة newborn genetic screening tools، والزراعة، وما إلى ذلك بناءً على أنواع مشاريع التعلم الآلي، يمكننا تحديد تطبيقات محددة. للتصنيف (التعلم الخاضع للإشراف Supervised Learning): تصنيف التسلسلات الأقصر shorter sequences إلى فئات classes (الشعبة phylum، والجنس genus، والأنواع species، وما إلى ذلك)؛ الاستدلال النشئي للتسلسلات phylogenetic inference of the sequences ؛

الكشف عن البلازميدات plasmids والكروموسومات chromosome. إيجاد مناطق الترميز coding regions؛ التنبؤ بالكروموسوم في الجينوميات البشرية؛ إلخ للتكتل clustering (التعلم غير الخاضع للإشراف Unsupervised Learning): تجميع كونتيكات الميتاجونومية binning of metagenomics contigs؛ تحديد البلازميدات والكروموسومات؛ يقرأ التكتل في الكروموسومات لتكتل أفضل؛ تجميع القراءات كمعالج أولي لتجميع القراءات، إلخ.

في هذا المشروع، سوف نفهم بنية تسلسل DNA / RNA / البروتين والتلاعب بها باستخدام مكتبات Python. سيوضح كيف يمكن استخدام خوارزميات التعلم الآلي لتصنيف تسلسل الحمض النووي DNA/ الحمض النووي الريبي RNA/ البروتين والتنبؤ به. سيتم توفير جدول مقارنة بين خوارزميات تصنيف التعلم الآلي التقليدية والحديثة لمجموعات البيانات الجينومية. سيتم تقديم مكتبة PyDNA بسيطة في Python لمعالجة سلاسل تسلسل DNA / RNA / البروتين وخوارزميات تصنيف التعلم الآلي.

2. تسلسل الحمض النووي

استنادًا إلى المعهد الوطني لبحوث الجينوم البشري national Human Genome Research Institute، فإن الحمض النووي الريبي منقوص الأكسجين Deoxyribonucleic Acid (DNA) هو مركب كيميائي يتكون من التعليمات اللازمة لتطوير وتوجيه أنشطة جميع الكائنات الحية تقريبًا. جزيء الحمض النووي هو هيكل حلزون مزدوج يتكون من خيطين مزدوجين ملتويين. يتكون الحمض النووي من أربع قواعد هي الأدينين [A] adenine، السيتوزين [C] cytosine، الجوانين guanine [G]، أو الثايمين [T] thymine. تسلسل الحمض النووي DNA sequence هو عملية مختبرية لتحديد تسلسل هذه القواعد الأربعة في جزيء الحمض النووي. يمكن العثور على مزيد من المعلومات حول تسلسل الحمض النووي في "مقدمة إلى تسلسل الحمض النووي".

دعونا نلقي نظرة على معالجة سلاسل تسلسل الحمض النووي DNA sequence strings manipulation باستخدام مكتبات Python. تم تنفيذ جميع استدعاءات الدوال المقدمة في مكتبة DNA مخصصة في Python تسمى PyDNA. مزيد من الشرح حول هذه المكتبة موجود لاحقًا في هذه المقالة.

3. التحقق من سلسلة تسلسل الحمض النووي

يجب أن تحتوي سلسلة تسلسل الحمض النووي على أربعة نيوكليوتيدات قاعدية four base nucleotides [" A"، "C"، "G"، "T""]. تسمح هذه الدالة بالتحقق من تسلسل الحمض النووي باستخدام أي قاعدة مخصصة محددة من النيوكليوتيدات.

مثال:

```
dna_sequence_string =
"ATATATCCCGGGAATTTTCGTAGTTAGGCTGATTTTATTGGCGCGAAAATTT"
is_dna_result = PyDNA.is_dna(dna_sequence_string)
print("DNA sequence string:\n{}".format(dna_sequence_string))
print("Is DNA:\n{}".format(is_dna_result))
```

النتائج:

```
DNA sequence string:
ATATATCCCGGGAATTTTCGTAGTTAGGCTGATTTTATTGGCGCGAAAATTT
Is DNA:
True
```

مثال:

```
dna_sequence_string =
"ATATATCCCGGGAATTTTCGTAGTTAGGCTGATTTTATTGGCGCGAAAATTF"
is_dna_result = PyDNA.is_dna(dna_sequence_string)
print("DNA sequence string:\n{}".format(dna_sequence_string))
print("Is DNA:\n{}".format(is_dna_result))
```

النتائج:

```
DNA sequence string:
ATATATCCCGGGAATTTTCGTAGTTAGGCTGATTTTATTGGCGCGAAAATTF
Is DNA:
False
```

4. عد النيوكليوتيدات القاعدية في السلسلة النصية لتسلسل الحمض النووي

تسمح الدالة أدناه بحساب نيوكليوتيدات تسلسل الحمض النووي DNA بأي قاعدة مخصصة محددة. سيتم إرجاع طول تسلسل الحمض النووي أيضًا.

مثال:

```
dna_sequence_string =
"ATATATCCCGGGAATTTTCGTAGTTAGGCTGATTTTATTGGCGCGAAAATTTTTT"
base_sequence_count, dna_sequence_length =
PyDNA.dna_count_nucleotide(dna_sequence_string,
base_sequence=["A","C","G","T"], is_length=True)
print("DNA sequence string:\n{}".format(dna_sequence_string))
print("DNA nucleotides count:\n{}".format(base_sequence_count))
print("DNA length:\n{}".format(dna_sequence_length))
```

النتائج:

```
DNA sequence string:
ATATATCCCGGGAATTTTCGTAGTTAGGCTGATTTTATTGGCGCGAAAATTTTTT
```

```
DNA nucleotides count:
{'A': 13, 'C': 7, 'G': 12, 'T': 23}
DNA length:
55
```

مثال:

```
dna_sequence_string =
"ATYTRTCCYGGYAATRYTCGTAGTTAGRCTGATYTTATTGGYGCGAARATTYYTR"base_sequence
_count, dna_sequence_length =
PyDNA.dna_count_nucleotide(dna_sequence_string,
base_sequence=["A","C","G","T","R","Y"], is_length=True)print("DNA
Nucleotide Count:\n{}".format(base_sequence_count))print("DNA
length:\n{}".format(dna_sequence_length))
```

النتائج:

```
DNA Nucleotide Count:
{'A': 10, 'C': 5, 'G': 10, 'T': 17, 'R': 5, 'Y': 8}
DNA length:
55
```

5. عكس سلسلة تسلسل الحمض النووي

تعمل الدالة التالية على عكس reverse السلسلة النصية string تسلسل الحمض النووي DNA.

مثال:

```
dna_sequence_string =
"ATATATCCCGGGAATTTTCGTAGTTAGGCTGATTTTATTGGCGCGAAAATTTTTT"
dna_reverse_sequence = PyDNA.dna_sequence_reverse(dna_sequence_string)
print("DNA Sequence String:\n{}".format(dna_sequence_string))
print("Reverse DNA sequence:\n{}".format(dna_reverse_sequence))
```

النتائج:

```
DNA Sequence String:
ATATATCCCGGGAATTTTCGTAGTTAGGCTGATTTTATTGGCGCGAAAATTTTTT
Reverse DNA sequence:
TTTTTTAAAAGCGCGGTTATTTTAGTCGGATTGATGCTTTTAAGGGCCCTATATA
```

6. استكمال سلسلة تسلسل الحمض النووي

يعتمد استكمال complementation تسلسل DNA على IUPAC Degeneracies Conversion.

مثال:

```
dna_sequence_test =
"ATATATCCCGGGAATTTTCGTAGTTAGGCTGATTTTATTGGCGCGAAAATTTTTT"
```

```
dna_complement_sequence =
PyDNA.dna_sequence_complement(dna_sequence_test)
print("DNA sequence string:\n{}".format(dna_sequence_test))
print("Complement DNA sequence:\n{}".format(dna_complement_sequence))
```

النتائج:

```
DNA sequence string:
ATATATCCCGGGAATTTTCGTAGTTAGGCTGATTTTATTGGCGCGAAAATTTTTT
Complement DNA sequence:
TATATAGGGCCCTTAAAAGCATCAATCCGACTAAAATAACCGCGCTTTTAAAAAA
```

7. عكس-استكمال السلسلة النصية تسلسل الحمض النووي

يتم تنفيذ الاستكمال العكسي reverse-complement لسلسلة النصية لتسلسل DNS باستخدام الدالتين dna_sequence_reverse() و t dna_sequence_complement().

مثال:

```
dna_sequence_string =
"ATATATCCCGGGAATTTTCGTAGTTAGGCTGATTTTATTGGCGCGAAAATTT"
dna_reverse_complement_sequence =
PyDNA.dna_sequence_reverse_complement(dna_sequence_string)
print("DNA sequence string:\n{}".format(dna_sequence_string))
print("Reverse-Complement DNA
sequence:\n{}".format(dna_reverse_complement_sequence))
```

النتائج:

```
ATATATCCCGGGAATTTTCGTAGTTAGGCTGATTTTATTGGCGCGAAAATTT
AAATTTTCGCGCCAATAAAATCAGCCTAACTACGAAAATTCCCGGGATATAT
```

8. عد محتوى GC في السلسلة النصية لتسلسل الحمض النووي

يمثل محتوى GC (GC-Content) النسبة المئوية للقواعد النيتروجينيةفي تسلسل الحمض النووي DNA أو الحمض النووي الريبي RNA.

مثال:

```
dna_sequence_string =
"ATATATCCCGGGAATTTTCGTAGTTAGGCTGATTTTATTGGCGCGAAAATTT"
gc_content = PyDNA.dna_count_gc_content(dna_sequence_string)
print("DNA sequence string:\n{}".format(dna_sequence_string))
print("GC-content:\n{}".format(gc_content))
```

النتائج:

```
DNA sequence string:
ATATATCCCGGGAATTTTCGTAGTTAGGCTGATTTTATTGGCGCGAAAATTT
```

```
GC-content:
36.5%
```

9. تحويل السلسلة النصية لتسلسل الحمض النووي إلى مصفوفة NumPy

تعد Python الرقمية (NumPy) مكتبة رئيسية للغة برمجة Python، مما يضيف دعمًا للمصفوفات والمصفوفات الكبيرة متعددة الأبعاد، جنبًا إلى جنب مع مجموعة كبيرة من الدوال الرياضية عالية المستوى للعمل على هذه المصفوفات. هذه المكتبة هي واحدة من أسرع المكتبات المتوفرةفي نظام بيانات Python اليوم.

يتم تجميع برنامج NumPy الأساسي وكود C المحسن جيداً. لهذا السبب يوصى بشدة باستخدامها في علم الأحياء الحاسوبي Computational Biology والمعلوماتية الحيوية Bioinformatics لمعالجة سلسلة عالية الأداء وخوارزميات التعلم الآلي. لا يزال العديد من علماء البيانات يستخدمون كائنات القائمة List / المجموعات Sets / القاموس Dictionary لمعالجة السلاسل النصية strings manipulationفي Python. يشتكي البعض منهم من أن لغة Python بطيئة جداً عندما يكتبون أكواد برمجة رديئةفي عملهم اليومي. أوصي دائمًا أصدقائي علماء او مهندسي البيانات بأخذ دورات متقدمةفي برمجة Python قبل كتابة سطر واحد من كود Python. عندما تسنح لك الفرصة، أود أن تقرأ ورقة المدونة التالية للمتعة: "إعادة هيكلة كود بايثون لمشاريع التعلم الآلي. Python "Spaghetti Code" Everywhere! "

يوجد أدناه كود دالة PyDNA لتحويل السلسلة النصية لتسلسل DNA إلى مصفوفة NumPy أحادية البعد.

```
@staticmethod
def dna_sequence_np_array(dna_sequence_string):
dna_sequence_array = None
try:
dna_sequence_string = dna_sequence_string.lower()
regex_acgt = re.compile('[^acgt]')
if (regex_acgt.search(dna_sequence_string) == None):
dna_sequence_array = np.array(list(dna_sequence_string))
else:
dna_sequence_array = None
except:
print(PyDNA.get_exception_info())
if PyDNA._app_is_log: PyDNA.write_log_file("error",
PyDNA.get_exception_info())
return dna_sequence_array
```

مثال:

```
dna_sequence_string =
"ATATATCCCGGGAATTTTCGTAGTTAGGCTGATTTTATTGGCGCGAAAATTTTTT"
```

```
dna_np_array = PyDNA.dna_sequence_np_array(dna_sequence_string)
print("DNA sequence string:\n{}".format(dna_sequence_string))
print("DNA NumPy array:\n{}".format(dna_np_array))
```

النتائج:

```
DNA sequence string:
ATATATCCCGGGAATTTTCGTAGTTAGGCTGATTTTATTGGCGCGAAAATTTTTT
DNA NumPy array:
['a' 't' 'a' 't' 'a' 't' 'c' 'c' 'c' 'g' 'g' 'g' 'a' 'a' 't' 't' 't'
't' 'c' 'g' 't' 'a' 'g' 't' 't' 'a' 'g' 'g' 'c' 't' 'g' 'a' 't' 't'
't' 't' 'a' 't' 't' 'g' 'g' 'c' 'g' 'c' 'g' 'a' 'a' 'a' 'a' 't' 't'
't' 't' 't' 't']
```

10. البحث عن أنماط تسلسل الحمض النووي

يعد البحث عن أنماط تسلسل الحمض النووي DNA Sequence Patterns مهمة قياسيةفي المعلوماتية الحيوية اليوم بمافي ذلك مجالات البروتين protein domains، وأشكال ربط عامل نسخ الحمض النووي DNA transcription factor binding motifs، ومواقع قطع إنزيم التقييد restriction enzyme cut sites، ومواقع تمهيدي PCR المتدهورة degenerate PCR primer sites، وتشغيل أحاديات النيوكليوتيدات runs of mononucleotides وغيرها الكثير. تحتوي مكتبة PyDNA على طريقة dna_sequence_pattern() بسيطة للعثور على أنماطفي السلاسل النصية لتسلسل الحمض النووي.

```
@staticmethod
def dna_sequence_pattern(dna_sequence_string, dna_sequence_pattern):
search_result = False
try:
search_pattern = re.search(dna_sequence_pattern.lower(),
dna_sequence_string.lower())
if search_pattern: search_result = True
except:
print(PyDNA.get_exception_info())
if PyDNA._app_is_log: PyDNA.write_log_file("error",
PyDNA.get_exception_info())
return search_result
```

دعونا نلقي نظرة على بعض الأمثلة.

مثال 1.

```
dna_sequence_string =
"ATATATCCCGGGAATTTTCGTAGTTAGGCTGATTTTATTGGCGCGAAAATTTTTT"
dna_sequence_pattern = "AATTTT"
result = PyDNA.dna_sequence_pattern(dna_sequence_string,
dna_sequence_pattern)
```

```
print(result)
True
```

مثال 2:

```
dna_sequence_string =
"ATATATCCCGGGAATTTTCGTAGTTAGGCTGATTTTATTGGCGCGAAAATTTTTT"
dna_sequence_pattern = "AATTTTAA"
result = PyDNA.dna_sequence_pattern(dna_sequence_string,
dna_sequence_pattern)
print(result)
False
```

11. ترجمة السلسلة النصية لتسلسل الحمض النووي إلى بروتين

لترجمة translate تسلسل الحمض النووي إلى بروتينات، يتم استخدام مخطط الشفرة الوراثية للحمض النووي DNA Genetic Code Chart.

مثال:

```
dna_sequence_string = "ATGGAAGTATTTAAAGCGCCACCTATTGGGATATAAG"
protein_translation =
PyDNA.dna_protein_translation(dna_sequence_string)
print("DNA sequence string:\n{}".format(dna_sequence_string))
print("Protein translation:\n{}".format(protein_translation))
```

النتائج:

```
DNA sequence string:
ATGGAAGTATTTAAAGCGCCACCTATTGGGATATAAG
Protein translation:
MEVFKAPPIGI
```

12. نسخ السلسلة النصية لتسلسل الحمض النووي إلى السلسلة النصية لتسلسل الحمض النووي الريبي

تقوم الدالة الموجودة أدناه بنسخ السلسلة النصية لتسلسل الحمض النووي DNA إلى السلسلة النصية string لتسلسل الحمض النووي الريبي RNA.

مثال:

```
dna_sequence_string = "ATGGAAGTATTTAAAGCGCCACCTATTGGGATATAAG"
rna_transcription = PyDNA.dna_rna_transcription(dna_sequence_string)
print("DNA sequence string:\n{}".format(dna_sequence_string))
print("RNA transcription:\n{}".format(rna_transcription))
```

النتائج:

```
DNA sequence string:
ATGGAAGTATTTAAAGCGCCACCTATTGGGATATAAG
RNA transcription:
AUGGAAGUAUUUAAAGCGCCACCUAUUGGGAUAUAAG
```

13. ترميز السلسلة النصية لتسلسل الحمض النووي

لاستخدام السلسلة النصية لتسلسل الحمض النوويفي مشاريع التعلم الآلي، يجب ترميزها أولاً. لا تعمل خوارزميات التعلم الآلي الرياضية مع البيانات النصية الفئوية text categorical data. اعتمادًا على خوارزمية التعلم الآلي المحددة، هناك ثلاثة أنواع رئيسية من ترميز السلسلة النصية لتسلسل الحمض النووي:

1. ترميز التسمية Label Encoding: هذا ترميز التسمية(العادي(ordinary سوف يقوم بترميز كل نوكليوتيد أساسي كقيمة عددية مخصصة. بشكل عام، لتكون أكثر دقة مع خوارزميات التعلم الآلي، يتم استخدام الأرقام العشرية floating (العشرية decimal). قبل تطبيق هذا الترميز، يجب تحويل سلسلة تسلسل الحمض النووي إلى مصفوفة NumPy أحادية البعد. هذا النوع من الترميز شائع جداً في التعلم الخاضع للإشراف باستخدام مكتبة scikit-Learn.في المثال أدناه، سيتم ترميز السلسلة النصية للتسلسل "ACGT" كـ [0.25، 0.5، 0.75، 1.0].

مثال:

```
dna_sequence_string =
"ATATATCCCGGGAATTTTCGTAGTTAGGCTGATTTTATTGGCGCGAAAATTTTTT"
dna_np_array = PyDNA.dna_sequence_np_array(dna_sequence_string)
dna_label_encoder = PyDNA.dna_label_encoder(dna_np_array)
print("DNA sequence string:\n{}".format(dna_sequence_string))
print("DNA NumPy array:\n{}".format(dna_np_array))
print("Custom Label Encoding:\n{}".format(dna_label_encoder))
```

النتائج:

```
DNA sequence string:
ATATATCCCGGGAATTTTCGTAGTTAGGCTGATTTTATTGGCGCGAAAATTTTTT
DNA NumPy array:
['a' 't' 'a' 't' 'a' 't' 'c' 'c' 'c' 'g' 'g' 'g' 'a' 'a' 't' 't' 't'
't' 'c' 'g' 't' 'a' 'g' 't' 't' 'a' 'g' 'g' 'c' 't' 'g' 'a' 't' 't'
't' 't' 'a' 't' 't' 'g' 'g' 'c' 'g' 'c' 'g' 'a' 'a' 'a' 'a' 't' 't'
't' 't' 't' 't']
Custom Label Encoding:
[0.25 1. 0.25 1. 0.25 1. 0.5 0.5 0.5 0.75 0.75 0.75 0.25 0.25 1. 1. 1.
1. 0.5 0.75 1. 0.25 0.75 1. 1. 0.25 0.75 0.75 0.5 1. 0.75 0.25 1. 1.
1. 1. 0.25 1. 1. 0.75 0.75 0.5 0.75 0.5 0.75 0.25 0.25 0.25 0.25 1. 1.
1. 1. 1. 1. ]
```

2. ترميز واحد ساخن One-hot Encoding: غالبًا ما يستخدم هذا الترميزفي الشبكات العصبية الاصطناعية Artificial Neural Networks (ANN). في كثير من الأحيان، يُطلق على ANN اسم التعلم العميق Deep Learning. التعلم العميق (المعروف أيضًا باسم التعلم المنظم العميق Deep Structured Learning) هو جزء من مجموعة أوسع من أساليب تعلم الآلة القائمة على ANN مع التعلم التمثيلي representation learning. وهو يتضمن البنى الرئيسية التالية: الشبكات العصبية العميقة Deep Neural Networks (DNN)، شبكات الاعتقاد العميق Deep Belief Networks (DBN)، الشبكات العصبية المتكررة Recurrent Neural Networks (RNN) والشبكات العصبية التلافيفية Convolutional Neural Networks (CNN). يمكن لمكتبتي scikit-Learn و Keras توفير تطبيق الترميز هذا. بالنسبة للقاعدة القياسية للنيوكليوتيدات، فإن سلسلة التسلسل "ACGT" ستكون مشفرة على أنها [[1. 0. 0. 0.] [0. 1. 0. 0.] [0. 0. 1. 0.] [0. 0. 0. 1.]] باستخدام مصفوفة NumPy ['a' c 'g' t '].

مثال:

```
dna_sequence_string =
"ATATATCCCGGGAATTTTCGTAGTTAGGCTGATTTTATTGGCGCGAAAATTTTTT"
dna_np_array = PyDNA.dna_sequence_np_array(dna_sequence_string)
dna_one_hot = PyDNA.dna_onehot_encoder(dna_np_array)
print("DNA sequence string:\n{}".format(dna_sequence_string))
print("DNA NumPy array:\n{}".format(dna_np_array))
print("DNA One-Hot Encoding with Scikit-Learn
framework:\n{}".format(dna_one_hot))
```

النتائج:

```
DNA sequence string:
ATATATCCCGGGAATTTTCGTAGTTAGGCTGATTTTATTGGCGCGAAAATTTTTT
DNA NumPy array:
['a' 't' 'a' 't' 'a' 't' 'c' 'c' 'c' 'g' 'g' 'g' 'a' 'a' 't' 't' 't'
't' 'c' 'g' 't' 'a' 'g' 't' 't' 'a' 'g' 'g' 'c' 't' 'g' 'a' 't' 't'
't' 't' 'a' 't' 't' 'g' 'g' 'c' 'g' 'c' 'g' 'a' 'a' 'a' 'a' 't' 't'
't' 't' 't' 't']
DNA One-Hot Encoding with Scikit-Learn framework:
[[1. 0. 0. 0.] [0. 0. 0. 1.] [1. 0. 0. 0.] [0. 0. 0. 1.] [1. 0. 0. 0.]
[0. 0. 0. 1.] [0. 1. 0. 0.] [0. 1. 0. 0.] [0. 1. 0. 0.] [0. 0. 1. 0.]
[0. 0. 1. 0.] [0. 0. 1. 0.] [1. 0. 0. 0.] [1. 0. 0. 0.] [0. 0. 0. 1.]
[0. 0. 0. 1.] [0. 0. 0. 1.] [0. 0. 0. 1.] [0. 1. 0. 0.] [0. 0. 1. 0.]
[0. 0. 0. 1.] [1. 0. 0. 0.] [0. 0. 1. 0.] [0. 0. 0. 1.] [0. 0. 0. 1.]
[1. 0. 0. 0.] [0. 0. 1. 0.] [0. 0. 1. 0.] [0. 1. 0. 0.] [0. 0. 0. 1.]
[0. 0. 1. 0.] [1. 0. 0. 0.] [0. 0. 0. 1.] [0. 0. 0. 1.] [0. 0. 0. 1.]
[0. 0. 0. 1.] [1. 0. 0. 0.] [0. 0. 0. 1.] [0. 0. 0. 1.] [0. 0. 1. 0.]
[0. 0. 1. 0.] [0. 1. 0. 0.] [0. 0. 1. 0.] [0. 1. 0. 0.] [0. 0. 1. 0.]
[1. 0. 0. 0.] [1. 0. 0. 0.] [1. 0. 0. 0.] [1. 0. 0. 0.] [0. 0. 0. 1.]
[0. 0. 0. 1.] [0. 0. 0. 1.] [0. 0. 0. 1.] [0. 0. 0. 1.] [0. 0. 0. 1.]]
```

يمكن تقديم نفس النتائج باستخدام مكتبة Keras.

```
dna_sequence_string =
"ATATATCCCGGGAATTTTCGTAGTTAGGCTGATTTTATTGGCGCGAAAATTTTTT"
dna_np_array = PyDNA.dna_sequence_np_array(dna_sequence_string)
dna_one_hot = PyDNA.dna_onehot_encoder_keras(dna_np_array)
print("DNA sequence string:\n{}".format(dna_sequence_string))
print("DNA NumPy array:\n{}".format(dna_np_array))
print("DNA One-Hot Encoding with Scikit-Learn
framework:\n{}".format(dna_one_hot))
```

3. عد K-mer (K-mer Counting): في المعلوماتية الحيوية ، k-mers هي تكرارات الطول الموجودةفي التسلسل البيولوجي biological sequence. عادة، يشير المصطلح k-mer إلى كل التسلسل التالي من الطول k، بحيث يحتوي تسلسل AGAT على أربعة مونومرات monomers (A ، G ، A ، T) ، ثلاثة 2mers- (AG, GA, AT) ، اثنان 3-mers (AGA و GAT) وواحد 4-mer (AGAT) - من تعريف k-mer. بشكل عام ، يسمح تحليل تسلسل إلى قطع ذات حجم ثابت k-mers بمعالجة السلسلة النصية بسرعة وسهولة. يتم تطبيق هذا بحكمةفي طريقة حقيبة الكلمات bag of words لمعالجة اللغة الطبيعية Natural Language Processing (NLP) لخوارزميات التعلم الآلي. سيتم تغطية هذه الطريقةفي الموضوع التالي.

```
dna_sequence_string =
"ATATATCCCGGGAATTTTCGTAGTTAGGCTGATTTTATTGGCGCGAAAATTT"
k_mer_list, k_mer_numpy_array = PyDNA.k_mer_words(dna_sequence_string,
k_mer_length=6)
print("DNA sequence string:\n{}".format(dna_sequence_string))
print("K-mer list:\n{}".format(k_mer_list))
print("K-mer array:\n{}".format(k_mer_numpy_array))
```

النتائج:

```
DNA sequence string:
ATATATCCCGGGAATTTTCGTAGTTAGGCTGATTTTATTGGCGCGAAAATTT
K-mer list:
['atatat', 'tatatc', 'atatcc', 'tatccc', 'atcccg', 'tcccgg', 'cccggg',
'ccggga', 'cgggaa', 'gggaat', 'ggaatt', 'gaattt', 'aatttt', 'attttc',
'ttttcg', 'tttcgt', 'ttcgta', 'tcgtag', 'cgtagt', 'gtagtt', 'tagtta',
'agttag', 'gttagg', 'ttaggc', 'taggct', 'aggctg', 'ggctga', 'gctgat',
'ctgatt', 'tgattt', 'gatttt', 'atttta', 'ttttat', 'tttatt', 'ttattg',
'tattgg', 'attggc', 'ttggcg', 'tggcgc', 'ggcgcg', 'gcgcga', 'cgcgaa',
'gcgaaa', 'cgaaaa', 'gaaaat', 'aaaatt', 'aaattt']
K-mer array:
['atatat' 'tatatc' 'atatcc' 'tatccc' 'atcccg' 'tcccgg' 'cccggg'
'ccggga' 'cgggaa' 'gggaat' 'ggaatt' 'gaattt' 'aatttt' 'attttc'
'ttttcg' 'tttcgt' 'ttcgta' 'tcgtag' 'cgtagt' 'gtagtt' 'tagtta'
'agttag' 'gttagg' 'ttaggc' 'taggct' 'aggctg' 'ggctga' 'gctgat'
'ctgatt' 'tgattt' 'gatttt' 'atttta' 'ttttat' 'tttatt' 'ttattg'
```

```
'tattgg' 'attggc' 'ttggcg' 'tggcgc' 'ggcgcg' 'gcgcga' 'cgcgaa'
'gcgaaa' 'cgaaaa' 'gaaaat' 'aaaatt' 'aaattt']
```

14. حقيبة الكلمات للسلسلة النصية لتسلسل الحمض النووي

تسلسل الحمض النووي هو عبارة عن بيانات نصية بسيطة غير منظمة unstructured text data. لهذا السبب، يجب أن تكون المعالجة اللغوية الطبيعية NLP أداة ممتازة لاستخدامها من أجل ذلك. الفكرة الرئيسية لاستخدام المعالجة اللغوية الطبيعية هي السماح لأجهزة الكمبيوتر بفهم النص غير المنظم واسترداد أجزاء ذات معنى من المعلومات لاتخاذ قرارات العمل. المعالجة اللغوية الطبيعية هو جزء من نظام الذكاء الاصطناعي.

حقيبة الكلمات Bag of words هي إحدى الطرق المستخدمةفي المعالجة اللغوية الطبيعية. وتتمثل مهمتها الرئيسيةفي تحويل بيانات النص الخام إلى كلمات وإحصاء تكرارهافي النص. ترتيب هذه الكلماتفي النص غير مناسب. بالنسبة للمستند النصي، يتم إنشاء مصفوفة من عدد الرموز المميزة token. بشكل عام، تمثل هذه المصفوفة متجه الميزات features vector النهائية التي سيتم تطبيقها في خوارزميات التعلم الآلي.في مثالنا أدناه، فإن قائمة k-mers لتكرارات الحمض النووي اللاحقة هي معلمة الإدخال لتوليد حقيبة الكلمات لها.

مثال:

```
k_mer_list = ['atatat', 'tatatc', 'atatcc', 'tatccc', 'atcccg',
'tcccgg', 'cccggg', 'ccggga', 'cgggaa', 'gggaat']
word_ngram = 1
k_mer_token_count = PyDNA.bag_of_word_list(k_mer_list, word_ngram)
print("K-mer list:\n{}".format(k_mer_list))
print("Word ngram:\n{}".format(word_ngram))
print("K-mer matrix token
counts:\n{}".format(k_mer_token_count.toarray()))
```

النتائج:

```
K-mer list:
['atatat', 'tatatc', 'atatcc', 'tatccc', 'atcccg', 'tcccgg', 'cccggg',
'ccggga', 'cgggaa', 'gggaat']
Word ngram:
1
K-mer matrix of token counts:
[[1 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 1 0 0]
[0 1 0 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 1 0]
[0 0 1 0 0 0 0 0 0 0]
[0 0 0 0 0 0 0 0 0 1]
[0 0 0 1 0 0 0 0 0 0]
[0 0 0 0 1 0 0 0 0 0]
```

```
[0 0 0 0 0 1 0 0 0 0]
[0 0 0 0 0 0 1 0 0 0]]
```

15. تسلسل الحمض النووي الريبي RNA

استنادًا إلى المعهد الوطني لبحوث الجينوم البشري، فإن Ribonucleic Acid (RNA) هو حمض نووي مشابهفي هيكله للحمض النووي DNA ولكنه يختلف بطرق خفية. تستخدم الخلية RNAفي عدد من المهام المختلفة، أحدها يسمى RNA المرسال messenger RNA أو mRNA. وهذا هو جزيء معلومات الحمض النووي الذي ينقل المعلومات من الجينوم إلى بروتينات عن طريق الترجمة translation. شكل آخر من أشكال الحمض النووي الريبي هو حمض نووي ريبوزي ناقل tRNA، أو transfer RNA، وهذه جزيئات الحمض النووي الريبي غير المشفرة للبروتين والتي تحمل فيزيائيًا الأحماض الأمينية amino acids إلى موقع الترجمة الذي يسمح بتجميعهافي سلاسل من البروتينات في عملية الترجمة.

يقوم تسلسل الحمض النووي الريبي بتحليل النسخة الخلوية المتغيرة باستمرار باستخدام تقنيات تسلسل الجيل التالي Next-Generation Sequencing (NGS). (NGS) هي عملية أتمتة واسعة النطاق تحقق التسلسل السريع لأزواج القواعدفي عينة DNA أو RNA. مكن NGS الباحثين من إجراء العديد من الدراسات البيولوجية ويمكنهم تسلسل جينوم بشري كاملفي يوم واحد. تُستخدم تطبيقات NGSفي مجموعة متنوعة من التقنيات الحديثة مثل تسلسل الجينوم الفيروسي الكامل عالي الإنتاجية high-throughput complete viral genome sequencing، واكتشاف تقلب جينوم الفيروس virus genome variability detection والتطورفي مضيف evolution in a host.

دعونا نلقي نظرة على بعض دوال PyDNA لدعم معالجة السلاسل النصية لتسلسل الحمض النووي الريبي RNA.

16. التحقق من صحة السلسلة النصية لتسلسل الحمض النووي الريبي

يجب أن تحتوي سلسلة تسلسل الحمض النووي الريبي على أربعة نيوكليوتيدات قاعدية [" "A"، "C"، "G"، ""U"]. تسمح هذه الدالة بالتحقق من تسلسل الحمض النووي الريبي RNA باستخدام أي نيوكليوتيدات أساسية مخصصة custom base nucleotides.

مثال:

```
rna_sequence_string =
"AUGGCCAUGGCGCCCAGAACUGAGAUCAAUAGUACCCGUAUUAACGGGUGA"
is_rna_result = PyDNA.is_rna(rna_sequence_string,
base_sequence=["A","C","G","U"])
print("RNA sequence string:\n{}".format(rna_sequence_string))
print("Is RNA:\n{}".format(is_rna_result))
```

النتائج:

```
RNA sequence string:
AUGGCCAUGGCGCCCAGAACUGAGAUCAAUAGUACCCGUAUUAACGGGUGA
Is RNA:
True
```

مثال:

```
rna_sequence_string =
"AUGGCCTUGGCGCCCAGAACUGAGAUCTAUAGUACCCGUAUUAACTGGUGA"
is_rna_result = PyDNA.is_rna(rna_sequence_string,
base_sequence=["A","C","G","U"])
print("RNA sequence string:\n{}".format(rna_sequence_string))
print("Is RNA:\n{}".format(is_rna_result))
```

النتائج:

```
RNA sequence string:
AUGGCCTUGGCGCCCAGAACUGAGAUCTAUAGUACCCGUAUUAACTGGUGA
Is RNA:
False
```

17. عد النيوكليوتيدات الأساسية في السلسلة النصية لتسلسل الحمض النووي الريبي

تتيح الدالة أدناه عد نيوكليوتيدات تسلسل الحمض النووي الريبي مع أي قاعدة مخصصة محددة. يتم إرجاع طول تسلسل RNA أيضًا.

مثال:

```
rna_sequence_string =
"AUGGCCAUGGCGCCCAGAACUGAGAUCAAUAGUACCCGUAUUAACGGGUGA"
base_sequence_count, rna_sequence_length =
PyDNA.rna_count_nucleotide(rna_sequence_string,
base_sequence=["A","C","G","U"], is_length=True)
print("RNA sequence string:\n{}".format(rna_sequence_string))
print("RNA nucleotides count:\n{}".format(base_sequence_count))
print("RNA length:\n{}".format(rna_sequence_length))
```

النتائج:

```
RNA sequence string:
AUGGCCAUGGCGCCCAGAACUGAGAUCAAUAGUACCCGUAUUAACGGGUGA
RNA nucleotides count:
{'A': 15, 'C': 12, 'G': 14, 'U': 10}
RNA le
```

18. ترجمة السلسلة النصية لتسلسل الحمض النووي الريبي إلى بروتين

لترجمة تسلسل الحمض النووي الريبي إلى بروتينات، يتم استخدام مخطط الشفرة Code Chart الجينية لـ RNA.

مثال:

```
rna_sequence_string                                                    =
"AUGGCCAUGGCGCCCAGAACUGAGAUCAAUAGUACCCGUAUUAACGGGUGA"
protein_translation                          =                      PyDNA.
rna_protein_translation(rna_sequence_string)
print("RNA      sequence      string:\n{}".format(rna_sequence_string))
print("Protein translation:\n{}".format(protein_translation))
```

النتائج:

```
RNA sequence string:
AUGGCCAUGGCGCCCAGAACUGAGAUCAAUAGUACCCGUAUUAACGGGUGA
Protein translation:
MAMAPRTEINSTRING
```

19. التراصف التسلسلي لحمض النووي والحمض النووي الريبي والبروتين

تعد التراصف التسلسلي Sequence alignment طريقة لترتيب تسلسل الحمض النووي أو الحمض النووي الريبي أو البروتين لتحديد مناطق التشابه التي قد تكون نتيجة للعلاقات الوظيفية أو الهيكلية أو التطورية بين التسلسلات. يتم تمثيل التسلسلات المتراصفة Aligned sequences لبقايا الأحماض الأمينية أو النيوكليوتيدات عادةً كصفوف داخل مصفوفة. يتم إدخال الفجوات بين البقايا بحيث يتم محاذاة الأحرف المتطابقة أو المتشابهةفي أعمدة متتالية. يُستخدم التراصف التسلسلي أيضًا للتسلسلات غير البيولوجية، مثل حساب تكلفة المسافة بين السلاسل في لغة طبيعية أوفي البيانات المالية.

تعد البرمجة الديناميكية (DP) Dynamic Programming إحدى التقنيات الأساسية المستخدمة اليوم للتراصف التسلسلي بشكل أسرع. DP هي طريقة تحسين رياضية وطريقة برمجة كمبيوتر. يشير في كلا السياقين إلى تبسيط مشكلة معقدة عن طريق تقسيمها إلى مشاكل فرعية أبسط بطريقة تكرارية. بينما لا يمكن تفكيك بعض مشكلات القرار بهذه الطريقة، غالبًا ما تتفكك القرارات التي تمتد على عدة نقاط زمنية بشكل متكرر. وبالمثل،في علوم الكمبيوتر، إذا كان من الممكن حل المشكلة على النحو الأمثل عن طريق تقسيمها إلى مشاكل فرعية ثم إيجاد الحلول المثلى للمشكلات الفرعية بشكل متكرر، فيقال إن لديها بُنية أساسية مثالية.

يمكن أن يكون للتراصف التسلسلي التطبيقات الرئيسية التالية: بالنظر إلى تسلسل جديد، توقع وظيفتها بناءً على التشابه مع تسلسل آخر، والعثور على مناطق جزيئية مهمة، وتحديد القيود التطوريةفي العمل،

والعثور على الطفراتفي مجموعة أو عائلة من الجينات، ويعطينا بيولوجيا الشبكة معلومات وظيفية عن الجينات / البروتينات، تحليل روابط الطفرات الجينات غير المعروفة بالأمراض، إلخ.

إذا قارنا تسلسلين، فإنه يُعرف باسم التراصف التسلسلي الزوجي pairwise sequence alignment. إذا قارنا أكثر من تسلسلين، فإنه يُعرف باسم التراصف التسلسلي المتعدد multiple sequence alignment. بشكل عام، هناك نوعان من التراصف التسلسلي: التراصف العام والمحلي global and local alignments. يعد التراصف العام، التي تحاول محاذاة كل بقايافي كل تسلسل، مفيدة للغاية عندما تكون التسلسلاتفي مجموعة الاستعلام متشابهة ومتساوية الحجم تقريبًا. (هذا لا يعني أن التراصف العام لا يمكن أن تبدأ و / أو تنتهيفي فجوات.) تقنية التراصف العام بشكل عام هي خوارزمية Needleman-Wunsch، والتي تعتمد على البرمجة الديناميكية. يعد التراصف المحلي أكثر فائدة للتسلسلات غير المتشابهة التي يشتبهفي احتوائها على مناطق متشابهة أو أشكال تسلسلية مماثلة ضمن سياق التسلسل الأكبر. تعد خوارزمية Smith-Waterman طريقة تراصف محلية عامة تستند إلى نفس مخطط البرمجة الديناميكي ولكن مع خيارات إضافية للبدء والانتهاءفي أي مكان.

يوجد أدناه كود لمقارنة تسلسلين باستخدام الخوارزميات العالمية والمحلية مع مكتبة PyDNA.

```
sequence_1 = "ACGGGT"
sequence_2 = "ACG"
print("Needleman-Wunsch Global Algorithm")
PyDNA..needleman_wunsch_global(sequence_1, sequence_2)
print("Smith-Waterman Local Algorithm")
PyDNA.smith_waterman_local(sequence_1, sequence_2)
```

النتائج:

```
Needleman-Wunsch Global Algorithm
score: 15
ACGGGT
AC G
AC — G-Smith-Waterman Local Algorithm
score: 30
ACG
```

20. نظرة عامة على مكتبة Biopython

مكتبة Python الأكثر شيوعًا المستخدمةفي علم الأحياء الحاسوبي والمعلوماتية الحيوية هي Biopython. مشروع Biopython عبارة عن مجموعة مفتوحة المصدر من أدوات Python غير التجارية للبيولوجيا الحاسوبية والمعلوماتية الحيوية، التي أنشأتها جمعية دولية للمطورين. يحتوي على فئات لتمثيل التسلسلات البيولوجية والتعليقات التوضيحية المتسلسلة، وهو قادر على القراءة والكتابة إلى مجموعة متنوعة من تنسيقات الملفات. كما يسمح بالوسائل البرمجية للوصول إلى قواعد البيانات

عبر الإنترنت للمعلومات البيولوجية، مثل تلك الموجودةفي NCBI. تعمل الوحدات المنفصلة على توسيع قدرات Biopython للتراصف العام sequence alignment، وبُنية البروتين protein structure، وعلم الوراثة السكانية population genetics، وعلم الوراثة phylogenetics، وتسلسلات الاشكال sequence motifs، والتعلم الآلي. Biopython هو واحد من عدد من مشاريع *Bio المصممة لتقليل تكرار الكود code duplicationفي علم الأحياء الحسابي.

للمقارنة، دعنا نستخدم هذه المكتبة للتراصف التسلسلي الموضحة أعلاه. كما ترون من البرنامج أدناه، تم استيراد مكتبة Bio.

```
from Bio import pairwise2
from Bio.pairwise2 import format_alignmentsequence_1 = "ACGGGT"
sequence_2 = "ACG"
print("Needleman-Wunsch Global Algorithm")
alignments = pairwise2.align.globalxx(sequence_1, sequence_2)
for item in alignments:
    print(format_alignment(*item))
print("Smith-Waterman Local Algorithm")
alignments = pairwise2.align.localxx(sequence_1, sequence_2)
for item in alignments:
    print(format_alignment(*item))
```

النتائج:

```
Needleman-Wunsch Global Algorithm
ACGGGT
||   |
AC--G-
  Score=3
ACGGGT
|| |
AC-G--
  Score=3
ACGGGT
|||
ACG---
  Score=3Smith-Waterman Local Algorithm
1 ACGGG
  ||   |
1 AC--G
  Score=3
1 ACGG
  || |
1 AC-G
  Score=3
1 ACG
  |||
```

```
1 ACG
  Score=3
```

كلا المكتبتين تنتج نفس النتائج. بالنسبة لحالتنا، فإن أفضل التراصف العالمي والمحلي هي AC – G_ و ACG.

21. مكتبة PyDNA المخصصة

أثناء البحث في تطبيقات خوارزميات التعلم الآلي لبيانات الجينوم، وجدت العديد من إجراءات البرمجة القياسية والعامة ومعالجة البيانات المتسلسلة غير متوفرة. لا توفر مكتبة Biopython نماذج ANN وGradient Boosting الحديثة المطلوبة مثل الشبكات العصبية التلافيفية (CNN)، والشبكات العصبية المتكررة (RNN)، واكس جي بوست (XGBoost)، وما إلى ذلك. ولهذا السبب، قررت إنشاء مكتبة Python بسيطة حتى أتمكن من إعادة استخدامه في عملي اليومي للمعلومات الحيوية في التعلم الآلي. بعد شهرين من العمل، أصبحت هذه المكتبة بسيطة وجيدة وكبيرة. قد تتوفر حزمة الإعداد الخاصة به في المستقبل. فيما يلي أساسيات التصميم الرئيسية.

1. سهل الاستخدام عن طريق نسخ ولصق ملفات pydna.py و ipydna.py في أي مشروع Python.
2. أساليب تعلم الآلة العامة والإجراءات المنطقية لتدفق عمل المشروع بأكمله.
3. معالجة الأخطاء وملف التكوين وتنفيذ رسائل السجل.
4. صيانة بسيطة وترقيات مستقبلية.
5. وحدة اختبارات تنفيذ المشروع.

فيما يلي مثال على كود لملف الواجهة العامة لمكتبة PyDNA.

```
from zope.interface import Interface
class IPyDNA(Interface):
def dna_sequence_np_array(dna_sequence_string):
"""
convert a dna sequence string to numpy one-dimensional array
dna_sequence_string: dna sequence string
return: numpy one-dimensional array
"""
pass
```

22. تصنيف تسلسل الحمض النووي باستخدام خوارزميات التعلم الآلي

هناك طلب كبير لتطبيق التعلم الآلي على تحليلات مجموعة بيانات الجينوم اليوم. لا تزال العديد من الأسئلة حول هذا الموضوع غير واضحة على الإطلاق. دعني أذكر بعضها:

- ما هو مشفر تسلسل الحمض النووي DNA sequence encoder الذي يجب استخدامه بناءً على نموذج التعلم الآلي المحدد؟
- ما هي نماذج التعلم الآلي التي يجب استخدامها لمجموعات بيانات الجينوم العام وكيفية تفسيرها؟
- كيفية تحسين المعلمات الفائقة hyperparameters لنموذج التعلم الآلي؟
- كيفية تطبيق معالجة اللغة الطبيعية NLP على تسلسل الحمض النووي كمجموعة بيانات نصية غير مهيكلة unstructured text dataset؟
- كيفية التعامل مع مجموعات بيانات التعبير الجيني عالية الأبعاد وغير المسماة high dimensional and unlabeled gene expression datasets؟
- ما هي الطريقة التي يجب استخدامها للتعامل مع مجموعات البيانات الجينومية genomic datasets ذات الفئات غير المتوازنة imbalanced classes؟
- ما المقاييس metrics التي يجب استخدامها للتحقق من صحة نموذج التعلم الآلي باستخدام مجموعات بيانات الجينوم؟
- كيف يتم الكشف عن نموذج التعلم الالي واستهلاكه باستخدام استدعاء واجهة برمجة تطبيقات API خدمات الويب؟
- كيفية تصميم وبناء جانب العميل و / أو تطبيق سطح المكتب لاستخدام نماذج التعلم الآلي هذهفي البحث الحقيقي و / أو بيئات الأعمال الإنتاجية؟

يعد اختيار نموذج التعلم الآلي وتحسين المعلمات الفائقة جزءًا من تدفق مشروع التعلم الآلي من الخطوات الموضحة أدناه:

- وصف المشروع والمواصفات Project description and specifications..
- تحميل البيانات Data loading.
- معالجة البيانات Data preprocessing.
- استكشاف البيانات والتصور Data exploration and visualization.
- هندسة الميزات واختزالها Features engineering and reduction.
- ترميز الميزات والتسميات Features and labels encoding.
- تقسيم بيانات الميزات والتسميات Features and labels data splitting.
- تحجيم الميزات والتسميات Features and labels scaling.
- اختيار النموذج وتحسين المعلمات الفائقة Model selection and hyperparameters optimization.
- التحقق من الصحة المتقاطع للنموذج Model cross validation.

- التنبؤ بالنموذج Model prediction.
- تحليل مقاييس أداء النموذج Model performance metrics analysis.
- نشر إنتاج النموذج باستخدام تكامل تطبيقات تكنولوجيا المعلومات لواجهة برمجة تطبيقات الويب Model production deployment using Web APIs IT application integration.
- جدول إعادة التدريب النموذجي وإعادة نشر ذكاء الأعمال واتخاذ القرارات Model retraining schedule and redeployment business intelligence and decisions making.

يعد تحديد هذا النموذج لمجموعة بيانات جينوم معينة قرارًا مهمًا لعلماء المعلومات الحيوية وعلماء الأحياء والأطباء.في العديد من حالات الاستخدام، يتم تحديد هذا التحديد، على سبيل المثال، بناءً على اتساق طول تسلسل الحمض النووي عبر مجموعة البيانات. يمكن استخدام خوارزميات التعلم الآلي التقليدية (الانحدار الخطي واللوجستي Linear and Logistics Regressions، وأشجار القرار Decision Trees، وآلة المتجهات الداعمة Support Vector Machines، والغابات العشوائية Random Forest، وخوارزميات التعزيز Boosting Algorithms، وشبكة بايزي Bayesian Network، وما إلى ذلك) مع أي طول لتسلسل الحمض النووي. تتطلب خوارزميات ANN الحديثة مثل CNN وRNN طول تسلسل DNA ثابتفي عمود مجموعة البيانات بأكمله. توفر مكتبة PyDNA دالة بسيطة لتحديد ما إذا كانت السلسلة النصية لتسلسل الحمض النووي المحددة تحتوي على طول موحد أم لا.

```
dna_is_same_length = PyDNA.dna_sequence_is_equal_length(X)
if dna_is_same_length == False:
print("DNA sequence length validation")
```

دعونا نلقي نظرة على حالة الاستخدام الأولى. لنفترض أننا بحاجة إلى بناء نموذج تصنيف يمكنه التنبؤ بعائلة جينية gene family بناءً على مجموعة بيانات تسلسل الحمض النووي البشري. يتم تصنيف الجينات إلى عائلات بناءً على النوكليوتيدات المشتركة أو تسلسل البروتين. عائلة الجينات هي مجموعة من عدة جينات متشابهة، تتكون من ازدواج جين أصلي واحد، وبشكل عام مع وظائف كيميائية حيوية مماثلة. ستستخدم هذه الحالة ملف "human_data.txt" يمكن تنزيله من موقع GitHub على الويب.

ستوفر لنا طريقة المعلومات الخاصة بـ Pandas DataFrame وصفًا كاملاً لمجموعة البيانات. يحتوي على عمودين "تسلسل sequence " و "فئة class " مع 4380 صفًا.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4380 entries, 0 to 4379
```

```
Data columns (total 2 columns):
# Column Non-Null Count Dtype
--- ------ -------------- -----
0 sequence 4380 non-null object
1 class 4380 non-null int64
dtypes: int64(1), object(1)
memory usage: 68.6+ KB
None
```

مثال على مجموعة البيانات:

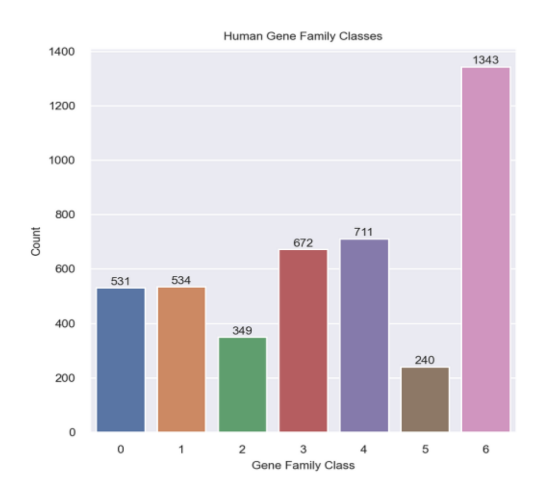
```
sequence class
0 ATGCCCCAACTAAATACTACCGTATGGCCCACCATAATTACCCCCA... 4
1 ATGAACGAAAATCTGTTCGCTTCATTCATTGCCCCCACAATCCTAG... 4
2 ATGTGTGGCATTTGGGCGCTGTTTGGCAGTGATGATTGCCTTTCTG... 3
3 ATGTGTGGCATTTGGGCGCTGTTTGGCAGTGATGATTGCCTTTCTG... 3
[4380 rows x 2 columns]
```

كما ترى، تحتوي مجموعة البيانات هذه على عمودي بيانات. عمود "التسلسل" هو السلسلة النصية لتسلسل الحمض النووي وعمود "الفئة" الذي يحتوي على سبع تسميات عائلة جينية محتملة موضحة في الجدول 1 أدناه.

Gene Family	Count	Class Label
G protein coupled receptors	531	0
Tyrosine kinase	534	1
Tyrosine phosphatase	349	2
Synthetase	672	3
Synthase	711	4
Ion channel	240	5
Transcription factor	1343	6

الجدول 1. اسم عائلة الجينات وتعدادها.

يوضح الشكل 1 المخطط الشريطي لتسميات الفئة class labels. لدينا حالة فئات غير متوازنة imbalanced classes مع مجموعة البيانات هذه. بشكل عام، قبل تطبيق خوارزميات التعلم الآلي، يجب أن تكون هذه الفئات عبارة عن زيادة في العينات oversampling أو نقص في العينات undersampling. دعنا نكتشف ما إذا كنا بحاجة إلى تطبيق أي من هذه التقنيات لنموذج التصنيف الخاص بنا.



الشكل 1. المخطط الشريطي لتسميات الطبقة البشرية.

فيما يلي أمثلة على تصنيفات الفئة y.

```
y class
0 4
1 4
2 3
3 3
```

إليك حقيبة الكلمات النهائية للميزات X وأمثلة أعمدة الفئة.

```
X class
(0, 52803) 1
(0, 207969) 1
(0, 136621) 1
(0, 79202) 1
```

دعونا نتحقق من الطول الموحد uniform length لتسلسل الحمض النووي.

```
X = PyDNA.select_df_column(df_dna, "sequence")
dna_is_same_length = PyDNA.dna_sequence_is_equal_length(X)
print(dna_is_same_length)
False
```

والنتيجة هي "False "، وبالتالي فإن طول تسلسل الحمض النووي ليس موحداً عبر العمود.في هذه الحالة، سيتم استخدام خوارزميات التعلم الآلي التقليدية.

استنادًا إلى مكتبة PyDNA، يتم عرض الكود الكامل لخوارزميات التعلم الآلي أدناه. لقد علقت على كل سطر من التعليمات البرمجية لأي شخص لفهم كيفية عمل هذا البرنامج.

```
import sys
import time
import os
os.system("cls")import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import xgboost as xgbfrom sklearn.feature_extraction.text import
CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,
confusion_matrix, accuracy_score
from sklearn.metrics import roc_auc_score, precision_score,
recall_score, f1_score
from sklearn.naive_bayes import MultinomialNB
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler, MinMaxScaler,
RobustScaler, MaxAbsScaler
from pydna import PyDNAimport warnings
warnings.filterwarnings('ignore')def
get_program_running(start_time):
    end_time = time. process_time()
    diff_time = end_time - start_time
    result = time.strftime("%H:%M:%S", time.gmtime(diff_time))
    print("program runtime: {}".format(result))def main():
    # text file path and name. this path to be defined in the
project config file
    human_data_txt = r"folder_path\human_data.txt"

    # data load
    df_dna = PyDNA.pandas_read_data("TXT", human_data_txt, None)#
select y label
    y = PyDNA.select_y_label(df_dna, "class")

    # generate a k-mer of words data frame column
    df_dna = PyDNA.create_dataframe_column_words(df_dna,
"sequence", "words")

    # show y label imbalanced classes plot
    PyDNA.imbalanced_classes_plot(y, True, "class", "Gene Family
Class", "Count", "Human Gene Family Classes")

    # generate X feature bag of works
    X = PyDNA.bag_of_word_series(df_dna["words"], 4)

    # data split in train, valid and test (80%/10%/10%)
    X_train, y_train, X_valid, y_valid, X_test, y_test =
PyDNA.train_validation_test_split(X, y, True, test_size=0.2,
valid_size=0.5)
```

```
    # create machine learning model and optimize it's
hyperparameters
    ml_model, ml_model_hyperparameter =
PyDNA.create_ml_model("MultinomialNB", X_train, y_train)
    print(ml_model_hyperparameter)

    # get y predicted valid
    y_predicted_valid = PyDNA.ml_model_predict(ml_model, X_valid)

    # calculate valid classification metrics
    accuracy_score_value, precision_value, recall_value,
f1_score_value, confusion_matrix_value, classification_report_value
= PyDNA.calculate_classification_metrics(y_valid,
y_predicted_valid)
    print("valid accuracy
score:\n{}\n".format(accuracy_score_value))
    print("valid precision:\n{}\n".format(precision_value))
    print("valid recall:\n{}\n".format(recall_value))
    print("valid f1 score:\n{}\n".format(f1_score_value))
    print("valid confusion
matrix:\n{}\n".format(confusion_matrix_value))
    print("valid classification
report:\n{}\n".format(classification_report_value))

    # get y predicted test
    y_predicted_test = PyDNA.ml_model_predict(ml_model, X_test)

    # calculate test classification metrics
    accuracy_score_value, precision_value, recall_value,
f1_score_value, confusion_matrix_value, classification_report_value
= PyDNA.calculate_classification_metrics(y_test, y_predicted_test)
    print("test accuracy
score:\n{}\n".format(accuracy_score_value))
    print("test precision:\n{}\n".format(precision_value))
    print("test recall:\n{}\n".format(recall_value))
    print("test f1 score:\n{}\n".format(f1_score_value))
    print("test confusion
matrix:\n{}\n".format(confusion_matrix_value))
    print("test classification
report:\n{}\n".format(classification_report_value))if __name__ ==
'__main__':
    start_time = time. process_time()
    main()
    get_program_running(start_time)
```

فيما يلي نتائج البرنامج باستخدام نموذج مصنف Multinomial Naive Bayes. تُظهر النتائج ستة مقاييس محسوبة رئيسية مستخدمةفي التحقق من صحة نماذج التصنيف: درجة الدقة accuracy score، الدقة precision، الاسترجاع recall، درجة f1 (f1 score)، مصفوفة الارتباك confusion matrix وتقرير التصنيف classification report. تعد نسبة 97.7٪ من درجات دقة الاختبار نتيجة

ممتازة لمجموعة البيانات الجينومية الخاصة بنا. بالنظر إلى قيم نقاط دقة التحقق والاختبار، يمكننا التأكد من أن مشكلة الضبط الزائد overfitting/ الضبط الناقص underfitting للنموذج ليست هي حالتنا. لهذه الأسباب، ليست هناك حاجة لتطبيق أي طرق فئات غير متوازنة لمجموعة البيانات الخاصة بنا.

```
validation accuracy score:
98.858validation precision:
98.872validation recall:
98.858validation f1 score:
98.86validation confusion matrix:
[[ 53   0   0   0   0   0   0]
 [  0  52   0   0   0   1   0]
 [  0   0  35   0   0   0   0]
 [  0   0   0  66   1   0   0]
 [  1   0   0   0  69   0   1]
 [  0   0   0   0   0  24   0]
 [  0   0   0   0   1   0 134]]validation classification report:
             precision    recall  f1-score   support0       0.98
1.00      0.99        53
          1       1.00      0.98      0.99        53
          2       1.00      1.00      1.00        35
          3       1.00      0.99      0.99        67
          4       0.97      0.97      0.97        71
          5       0.96      1.00      0.98        24
          6       0.99      0.99      0.99       135    accuracy
0.99       438
   macro avg       0.99      0.99       0.99       438
weighted avg       0.99      0.99       0.99       438test accuracy
score:
97.717test precision:
97.8test recall:
97.717test f1 score:
97.732test confusion matrix:
[[ 50   0   0   0   3   0   0]
 [  0  53   0   0   0   0   1]
 [  0   0  35   0   0   0   0]
 [  0   0   0  67   0   0   0]
 [  0   1   0   0  70   0   0]
 [  0   0   0   0   0  23   1]
 [  0   0   0   0   1   3 130]]test classification report:
             precision    recall  f1-score   support0       1.00
0.94      0.97        53
          1       0.98      0.98      0.98        54
          2       1.00      1.00      1.00        35
          3       1.00      1.00      1.00        67
          4       0.95      0.99      0.97        71
          5       0.88      0.96      0.92        24
          6       0.98      0.97      0.98       134    accuracy
0.98       438
```

```
   macro avg       0.97      0.98      0.97       438
weighted avg       0.98      0.98      0.98       438
```

يوضح الجدول 2 نتائج تطبيق أنواع مختلفة من نماذج تصنيف التعلم الآلي. تم الحصول على أفضل النتائج باستخدام نماذج المصنف Multinomial Naive Bayes وMulti-layer Perceptron مع أكثر من 97٪ من درجات دقة الاختبار. قدمت نماذج الانحدار اللوجستي Logistic Regression والغابات العشوائية Random Forest حوالي 92٪ من درجات دقة الاختبار. عادةً ما يوفر نموذج الغابات العشوائية نتائج جيدة، عمليًا، للعديد من مجموعات البيانات الممكنة. أوصي بالبدء بنموذج الغابات العشوائية لأي مشاريع تصنيف وانحدار تعلم آلي. لم تنجح نماذج تعزيز التدرج Gradient boosting modelsفي مجموعة البيانات لدينا. ربما، بشكل عام، هذه النماذج ليست جيدة بما يكفي لتصنيف مجموعات بيانات الجينوم. سأقدم المزيد من المعلومات والنتائج الجديدة حول هذا الموضوعفي أوراق المدونة القادمة.

Classification Model	Validation Accuracy Score, %	Test Accuracy Score, %
Logistic Regression LogisticRegression()	92.2	94.0
Decision Tree DecisionTreeClassifier()	76.4	81.7
Random Forest RandomForestClassifier()	92.2	92.4
C-Support Vector Classification SVC()	89.4	90.8
Multinomial Naive Bayes MultinomialNB()	98.8	97.7
Gradient Boosting GradientBoostingClassifier()	81.9	83.7
AdaBoost AdaBoostClassifier()	42.6	43.8
K-Nearest Neighbors KNeighborsClassifier()	78.3	81.7
Multi-layer Perceptron MLPClassifier()	98.8	97.4
Extreme Gradient Boosting XGBClassifier()	76.0	78.5

الجدول 2. نتائج نماذج مصنفات التعلم الآلي مع مجموعة بيانات عائلة الجينات.

23. تطبيق الشبكات العصبية التلافيفية (CNN) لتصنيف تسلسل الحمض النووي

في حالة الاستخدام الثانية، ستتوقع ما إذا كان تسلسل الحمض النووي يمكن أن يرتبط بالبروتين أم لا. البروتينات المرتبطة بالحمض النووي DNA-binding proteins هي بروتينات لها مجالات ربط الحمض النووي وبالتالي لها تقارب affinity محدد أو عام للحمض النووي أحادي أو مزدوج الشريطة single- or double-stranded DNA. مجال ربط الحمض النووي هو مجال بروتين مطوي بشكل مستقل يحتوي على شكل هيكلي واحد على الأقل يتعرف على الحمض النووي المزدوج أو المفرد احادي الخيط double- or single-stranded DNA. هذا سؤال جينومي وظيفي قياسي للكشف عن مواقع ربط عامل النسخفي تسلسل الحمض النووي.

الشبكات العصبية التلافيفية Convolutional Neural Networks (CNN) هي فئة من الشبكات العصبية العميقة Deep Neural Networks (DLN)، وهي الأكثر استخدامًا لتحليل الصور المرئية visual imagery. تُعرف أيضًا باسم shift invariant or space Invariant Artificial Neural Networks (SIANN)، بناءً على بنية الأوزان المشتركة shared-weights architecture وخصائص الترجمة الثابتة translation invariance characteristics. لديهم تطبيقاتفي التعرف على الصور والفيديو image and video recognition، وأنظمة التوصية recommender systems، وتصنيف الصور image classification، وتحليل الصور الطبية medical image analysis، ومعالجة اللغة الطبيعية natural language processing، وواجهات الدماغ_الحاسوب brain-computer interfaces، والسلاسل الزمنية المالية financial time series ، إلخ.

من المعروف أن CNN تستخدم بشكل عام لتحليل تلافيفات convolutions الصورة ثنائية الأبعاد (2D) وثلاثية الأبعاد (3D). من خلال مجموعة البيانات الجينومية الخاصة بنا، سيتم تطبيق نموذج بسيط 1D CNN ومكتبة Keras. يمكن تنزيل مثال على تسلسل الحمض النووي وبيانات نص التسمية label text data من روابط URL التالية:

```
dna_sequence = "https://raw.githubusercontent.com/abidlabs/deep-learning-genomics
-primer/master/sequences.txt"
dna_label = "https://raw.githubusercontent.com/abidlabs/deep-learning-genomics-
primer/master/labels.txt"
```

في المشاريع التفاعلية للتعلم الآلي باستخدام Jupyter Notebooks، على سبيل المثال، سيستغرق تحميل هذه البياناتفي إطار بيانات DataFrame لـ pandas بعض وقت تشغيل البرنامج. لتجنب ذلك، تم تطوير دالة عامة بسيطةفي مكتبة PyDNA. تقوم هذه الدالة بتحميل هذين الرابطين بالتوازي

وإنشاء ملف CSV نهائي بسلسلة تسلسل الحمض النووي وأعمدة رقم التسمية. المعلمة الثالثة "dna_sequence_protein" هي اسم معرف لملف CSV. تظهر استدعاء الدالة أدناه.

```
PyDNA.GenerateCSVFileParallel(dna_sequence, dna_label,
''dna_sequence_protein'')
```

يعد تحميل البيانات في مشاريع التعلم الآلي مهمة معنية خاصة في منطقة مجال البيانات الضخمة Big Data domain. حتى مجموعة البيانات البسيطة التي تحتوي على ملايين الصفوف ستبطئ عملية تحميل إطار بيانات pandas. سيؤدي تطبيق تقنيات مثل Python غير المتزامن وبرمجة المعالجة المتعددة إلى تحسين هذه العملية إلى حد كبير.

يعرض الجدول 3 عشرة صفوف من ملف "dna_sequence_protein.csv" النهائي. تحتوي أعمدة "dna_label" على قيمتين ثنائيتين: 0 – تسلسل الحمض النووي لا يمكن أن يرتبط بالبروتين و1 – يمكن أن يرتبط بالبروتين. هذا هو تصنيف ثنائي بسيط مهمة للتعلم الآلي.

dna_sequence	dna_label
CCGAGGGCTATGGTTTGGAAGTTAGAACCCTGGGGCTTCTCGCGGACACC	0
GAGTTTATATGGCGCGAGCCTAGTGGTTTTTGTACTTGTTTGTCGCGTCG	0
GATCAGTAGGGAAACAAACAGAGGGCCCAGCCACATCTAGCAGGTAGCCT	0
GTCCACGACCGAACTCCCACCTTGACCGCAGAGGTACCACCAGAGCCCTG	1
GGCGACCGAACTCCAACTAGAACCTGCATAACTGGCCTGGGAGATATGGT	1
AGACATTGTCAGAACTTAGTGTGCGCCGCACTGAGCGACCGAACTCCGAC	1
CCCGGCGAAGGCTGACGAATCCTCGACCGAACTCCAGTGAAGCCAACCGG	1
AGGCAGGTGGTCGTACAATGTTTTCGAAGAGATAGGGGGCCAGAGGCCTC	0
TACTGCCTATAGCGAAGAGCGCGAGAGGTATATCGAAGAATACCGAGCAA	0
CGTATCTTCGTGTGCTCTCCTTTAGAACTGCATCTCTAGAGTCAGAGAGG	0

الجدول 3. صفوف بيانات "dna_sequence_protein.csv".

للاطلاع على وصف مجموعة البيانات، تم تطبيق طريقة معلومات اطار بيانات Pandas. تحتوي مجموعة البيانات هذه على 2000 صف مع عمودين سلاسل "dna_sequence" وأرقام ثنائية "dna_label" من 0 و 1.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2000 entries, 0 to 1999
Data columns (total 2 columns):
# Column Non-Null Count Dtype
--- ------ -------------- -----
0 dna_sequence 2000 non-null object
1 dna_label 2000 non-null int64
dtypes: int64(1), object(1)
memory usage: 46.9+ KB
None
```

قبل تطبيق أي نماذج ANN، يجب التحقق من طول تسلسل الحمض النووي للتأكد من انتظامه uniformity. نتيجة الكود أدناه هي "True"، لذلك يمكن تطبيق نماذج CNN و RNN على مجموعة البيانات الجينومية هذه.

```
X = PyDNA.select_df_column(df_genomics, "dna_sequence")
dna_is_same_length = PyDNA.dna_sequence_is_equal_length(X)
print(dna_is_same_length)
```

تم ترميز تسلسل الحمض النووي DNA sequences وتسميات الفئة class labels باستخدام ترميز واحد ساخن one-hot encoding. كما تم توضيحه من قبل، يعد هذا أحد استخدامات الترميز القياسية الرئيسيةفي طرازي CNN وRNN. يظهر أدناه مثال لترميز واحد ساخن لتسلسل الحمض النووي (ميزات X).

```
[[[0. 1. 0. 0.]
  [0. 1. 0. 0.]
  [0. 0. 1. 0.]
  ...
  [1. 0. 0. 0.]
  [0. 1. 0. 0.]
  [0. 1. 0. 0.]][[0. 0. 1. 0.]
  [1. 0. 0. 0.]
  [0. 0. 1. 0.]
  ...
  [0. 0. 0. 1.]
  [0. 1. 0. 0.]
  [0. 0. 1. 0.]][[0. 0. 1. 0.]
  [1. 0. 0. 0.]
  [0. 0. 0. 1.]
  ...
  [0. 1. 0. 0.]
  [0. 1. 0. 0.]
  [0. 0. 0. 1.]]]
```

هنا تسمية الفئة ترميز واحد ساخن (تسمية y).

```
[[1. 0.]
[1. 0.]
[1. 0.]
...
[1. 0.]
[0. 1.]
[0. 1.]]
```

يوجد أدناه كود البرنامج الكامل لتنفيذ نموذج CNN لمجموعة البيانات الجينومية هذه باستخدام مكتبة PyDNA المخصصة.

```
import sys
import time
import os
os.system("cls")
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'import tensorflow as tf
tf.random.set_seed(10)import tensorflow.keras.backend as K
from tensorflow.keras.models import load_model
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import requests
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,
confusion_matrix, accuracy_score
from sklearn.metrics import roc_auc_score, precision_score,
recall_score, f1_score
from pydna import PyDNAimport warnings
warnings.filterwarnings("ignore")def
get_program_running(start_time):
    end_time = time.process_time()
    diff_time = end_time - start_time
    result = time.strftime("%H:%M:%S", time.gmtime(diff_time))
    print("program runtime: {}".format(result))

def main():
    # csv file path and name
    csv_path_file = r"csv_file_path/name.csv"# data frame load
    df_genomics = PyDNA.pandas_read_data("CSV", csv_path_file,
None)# remove rows and columns with missing values.
    df_genomics.dropna(how="all", inplace=True)# select X features
    X = PyDNA.select_df_column(df_genomics, "dna_sequence")# check
if dna sequences have the same legnth - part of dna sequence
preprocessing!
    dna_is_same_length = PyDNA.dna_sequence_is_equal_length(X)
    if dna_is_same_length == False:
        exit()# X features one-hot encoder
    X = PyDNA.cnn_X_onehot_encoder(X)# select y label
    y = PyDNA.select_df_column(df_genomics, "dna_label")# show y
label imbalanced classes plot
    PyDNA.imbalanced_classes_plot(y, True, "dna_label", "DNA bind
to protein class", "Count", "DNA Sequence Protein Classes")# y
label one-hot encoder
    y = PyDNA.cnn_y_onehot_encoder(y)# data split in train, valid
and test (80%/10%/10%)
    X_train, y_train, X_valid, y_valid, X_test, y_test =
PyDNA.train_validation_test_split(X, y, test_size=0.2,
valid_size=0.5)# create cnn model and get loss/metrics values
history
    epochs = 50
```

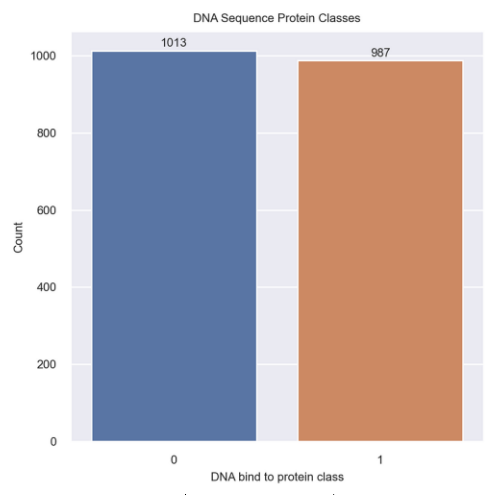
```
    data_split = 0.2
    cnn_model, cnn_history = PyDNA.create_cnn_model(y_train,
X_train, epochs, data_split)# plot cnn model loss
    font_size = 8
    PyDNA.cnn_model_loss_plot(cnn_history, font_size, "CNN Model
Loss", "Epoch", "Loss", ["Train", "Validation"])# plot cnn model
accuracy
    PyDNA.cnn_model_accuracy_plot(cnn_history, font_size, "CNN
Model Accuracy", "Epoch", "Accuracy", ["Train",
"Validation"])print("Model Validation")
    # get y_predicted valid
    y_predicted = cnn_model.predict(X_valid)# get max indices of
the maximum values along an axis
    y_val_max = PyDNA.get_max_nparray(y_valid)
    y_predicted_max = PyDNA.get_max_nparray(y_predicted)# calculate
valid classification metrics
    accuracy_score_value, precision_value, recall_value,
f1_score_value, confusion_matrix_value, classification_report_value
= PyDNA.calculate_classification_metrics(y_val_max,
y_predicted_max)
    print("valid accuracy
score:\n{}\n".format(accuracy_score_value))
    print("valid precision:\n{}\n".format(precision_value))
    print("valid recall:\n{}\n".format(recall_value))
    print("valid f1 score:\n{}\n".format(f1_score_value))
    print("valid confusion
matrix:\n{}\n".format(confusion_matrix_value))
    print("valid classification
report:\n{}\n".format(classification_report_value))print("Model
Test")
    # get y_predicted test
    y_predicted = cnn_model.predict(X_test)

    # get max indices of the maximum values along an axis
    y_test_max = PyDNA.get_max_nparray(y_test)
    y_predicted_max = PyDNA.get_max_nparray(y_predicted)# calculate
test classification metrics
    accuracy_score_value, precision_value, recall_value,
f1_score_value, confusion_matrix_value, classification_report_value
= PyDNA.calculate_classification_metrics(y_test_max,
y_predicted_max)
    print("test accuracy
score:\n{}\n".format(accuracy_score_value))
    print("test precision:\n{}\n".format(precision_value))
    print("test recall:\n{}\n".format(recall_value))
    print("test f1 score:\n{}\n".format(f1_score_value))
    print("test confusion
matrix:\n{}\n".format(confusion_matrix_value))
    print("test classification
report:\n{}\n".format(classification_report_value))
```

```
    # save cnn model h5
    cnn_model_path = r"cnn_model_path  "
    cnn_model_name = "cnn_model_name"
    PyDNA.cnn_model_save_h5(cnn_model, cnn_model_path,
cnn_model_name)

    # load cnn model h5
    cnn_model = PyDNA.cnn_model_load_h5(cnn_model_path,
cnn_model_name)
    print(cnn_model)if __name__ == '__main__':
    start_time = time.process_time()
    main()
    get_program_running(start_time)
```

من أول الأشياء التي يجب القيام بهافي مشاريع تصنيف التعلم الآلي هو التحقق من الفئات غير المتوازنة imbalanced classes كما فعلنا مع مجموعة بيانات عائلة الجينات السابقة. يوضح الشكل 2 مخطط شريطي لتسميات الطبقة الثنائية للحمض النووي. من الواضح أن كلا تصنيفي الفئةفي حالة توازن جيد.



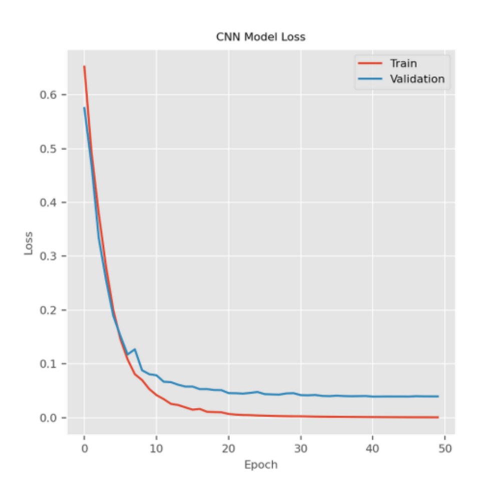
الشكل 2. رسم بياني شريطي لتسميات الطبقة الثنائية للحمض النووي.

نتائج البرنامج موضحة أدناه. نسبة 97.0٪ من درجة دقة الاختبار جيدة جداً. يمكن تطبيق نموذج CNN هذا لتصنيف تسلسل الحمض النووي الذي يمكن أن يرتبط بالبروتين أم لا.

```
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_1 (Conv1D)            (None, 39, 32)            1568
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 9, 32)             0
_________________________________________________________________
flatten_1 (Flatten)          (None, 288)               0
_________________________________________________________________
dense_1 (Dense)              (None, 16)                4624
```

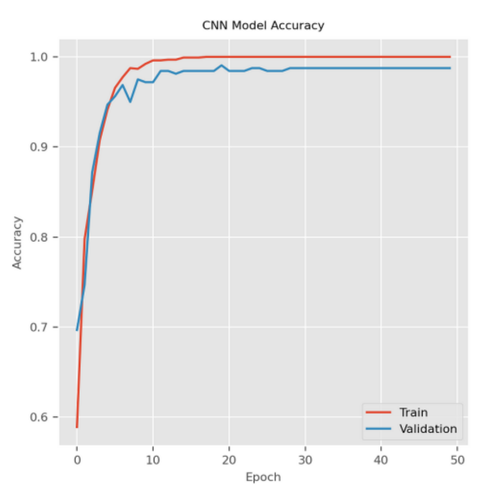
```
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 34
=================================================================
Total params: 6,226
Trainable params: 6,226
Non-trainable params: 0
_________________________________________________________________va
lidation accuracy score:
97.5validation precision:
97.544validation recall:
97.5validation f1 score:
97.5validation confusion matrix:
[[98  4]
 [ 1 97]]validation classification report:
              precision    recall  f1-score   support           0
0.99      0.96      0.98       102
           1       0.96      0.99      0.97        98    accuracy
0.97       200
   macro avg       0.98      0.98      0.97       200
weighted avg       0.98      0.97      0.98       200test accuracy
score:
97.0test precision:
97.019test recall:
97.0test f1 score:
97.0test confusion matrix:
[[97  4]
 [ 2 97]]test classification report:
              precision    recall  f1-score   support           0
0.98      0.96      0.97       101
           1       0.96      0.98      0.97        99    accuracy
0.97       200
   macro avg       0.97      0.97      0.97       200
weighted avg       0.97      0.97      0.97       200
```

إنها ممارسة جيدة عند استخدام CNNفي التعلم الآلي لتصور مخططات التحقق من صحة الخطأ والدقة لمجموعات التدريب train / التحقق من الصحة validation استنادًا إلى تكرارات الفترات epoch. عدد الفترات number of epochs هو نموذج معلمة فائقة hyperparameter يحدد عدد المرات التي ستعمل فيها خوارزمية التعلم من خلال مجموعة بيانات التدريب بأكملها. يوضح الشكل 3 مخطط خطأ التدريب / التحقق من صحة نموذج CNN (منحنى curve). بمجرد أن تتوقف خطأ مجموعة التحقق عن التحسن أو تزداد سوءًا خلال دورات التعلم، فقد حان الوقت لإيقاف التدريب لأن النموذج قد تقارب converged بالفعل وقد يكون مجرد ضبط زائد overfitting.في حالتنا، يجب أن يكون عدد الفترات بين 40 و 50. يجب ألا يكون هذا الرقم أقل من 40 لمنع الضبط الناقص النموذج underfitting. لذا، فإن اختيار عدد الفترات مهم جداً لضمان درجة دقة نموذج CNN.



الشكل 3. مخطط خطأ التدريب / التحقق من صحة لنموذج CNN.

يوضح الشكل 4 مخطط دقة تدريب / التحقق من الصحة لنموذج CNN. كما ترى، بعد 30 فترة، تكون دقة مجموعة التحقق مستقرة. لذلك، من خلال الجمع بين هاتين المخططين، يمكننا أن نستنتج أن اختيار عدد الفترات بين 40 و 50 هو حل جيد لمجموعة بيانات الجينوم المحددة هذه.



الشكل 4. مخطط دقة تدريب / التحقق من الصحة لنموذج CNN.

24. تطبيق شبكات الذاكرة طويلة قصيرة المدى (LSTM) لتصنيف تسلسل الحمض النووي

الشبكة العصبية المتكررة (RNN) Recurrent Neural Network هي فئة من الشبكات العصبية الاصطناعية (ANN) Artificial Neural Networks حيث تشكل الاتصالات بين العقد رسمًا بيانيًا موجهًا على طول تسلسل زمني temporal sequence. هذا يسمح لها بإظهار السلوك الديناميكي

الزمني temporal dynamic behavior. المشتقة من الشبكات العصبية امامية التغذية feedforward neural networks، يمكن لشبكات RNN استخدام حالتها الداخلية internal state (الذاكرة memory) لمعالجة تسلسلات متغيرة الطول من المدخلات. هذا يجعلها قابلة للتطبيق على مهام مثل التعرف على خط اليد handwriting recognition غير المقسم أو المتصل أو التعرف على الكلام speech recognition.

الذاكرة طويلة قصيرة المدى Long Short-Term Memory (LSTM) هي بنية RNN المستخدمةفي مجال التعلم العميق. على عكس الشبكات العصبية ذات التغذية الأمامية القياسية، فإن LSTM لديها اتصالات تغذية مرتدة feedback connections. لا يمكنها فقط معالجة نقاط البيانات الفردية (مثل الصور)، ولكن أيضًا التسلسل الكامل للبيانات (مثل الكلام أو الفيديو). على سبيل المثال، يمكن تطبيق LSTM على مهام مثل التعرف على خط اليد handwriting recognition غير المقسم والمتصل والتعرف على الكلام speech recognition واكتشاف الانحراف anomaly detection في حركة مرور الشبكة أو أنظمة كشف التسلل IDSs.

LSTMs قادرة على تعلم التبعيات طويلة المدى long-term dependencies مع حفظ العديد من الخطوات السابقةفي تسلسل مما يوفر بيانات كافية متاحة. يبشر LSTM بأي مهمة معالجة متسلسلة قد يوجد فيها تحلل هرمي محتمل، لكن لا تعرف مسبقًا ما هو هذا التحلل decomposition. إذا حددنا الحمض النووي كسلسلة ذات ذاكرة طويلة تتجلى من خلال الارتباطات طويلة المدى long-range correlations على طول التسلسل، فلماذا لا نطبق خوارزمية LSTM لتصنيف تسلسل الحمض النووي.

لتشغيل خوارزمية LSTM، التغييرات التالية مطلوبةفي البرنامج السابق باستخدام CNN.

```
# create lstm model and get loss/metrics values history
epoch = 50
data_split = 0.2
lstm_model, lstm_history = PyDNA.create_lstm_model(y_train,
X_train, epoch, data_split)# plot lstm model loss
font_size = 8
PyDNA. lstm_model_loss_plot(lstm_history, font_size, "LSTM Model
Loss", "Epoch", "Loss", ["Train", "Validation"])# plot lstm model
accuracy
PyDNA.lstm_model_accuracy_plot(lstm_history, font_size, "LSTM Model
Accuracy", "Epoch", "Accuracy", ["Train", "Validation"])
```

بالنسبة إلى ملف "dna_sequence_protein.csv" نفسه، تظهر النتائج أدناه.

```
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_1 (Conv1D)            (None, 39, 32)            1568
```

```
_________________________________________________________________
lstm_1 (LSTM)                (None, 39, 64)            24832
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 9, 64)             0
_________________________________________________________________
flatten_1 (Flatten)          (None, 576)               0
_________________________________________________________________
masking_1 (Masking)          (None, 576)               0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                36928
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 130
=================================================================
Total params: 63,458
Trainable params: 63,458
Non-trainable params: 0
_________________________________________________________________

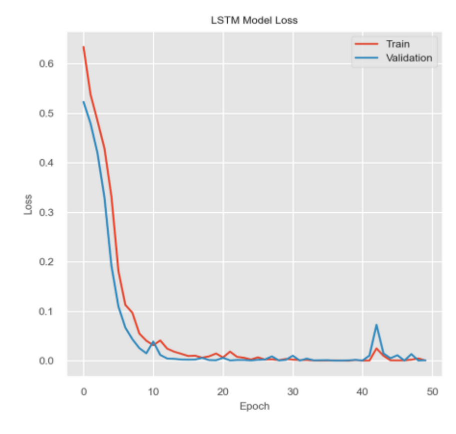
validation accuracy score:
98.5validation precision:
98.545validation recall:
98.5validation f1 score:
98.5validation confusion matrix:
[[99  3]
 [ 0 98]]validation classification report:
              precision    recall  f1-score   support           0
1.00      0.97      0.99       102
           1       0.97      1.00      0.98        98    accuracy
0.98       200
   macro avg       0.99      0.99      0.98       200
weighted avg       0.99      0.98      0.99       200test accuracy
score:
99.5test precision:
99.505test recall:
99.5test f1 score:
99.5test confusion matrix:
[[100   1]
 [  0  99]]test classification report:
              precision    recall  f1-score   support           0
1.00      0.99      1.00       101
           1       0.99      1.00      0.99        99    accuracy
0.99       200
   macro avg       0.99      1.00      0.99       200
weighted avg       1.00      0.99      1.00       200
```

عند تقييم أداء النموذج على بيانات الاختبار، كانت درجة الدقة التي تم الحصول عليها 99.5٪. لقد قمت بتشغيل مجموعتين من مجموعات البيانات الجينومية وخوارزميات LSTM توفر أفضل النتائج لتصنيف تسلسل الحمض النووي حتى الآن. إذا طبقنا خوارزمية مصنف بيربيسترون Perceptron

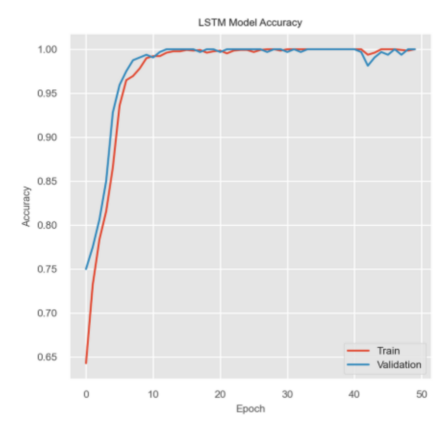
متعدد الطبقات MLPClassifier()، فيمكن عرض النتائج النهائية أدناه. درجة دقة الاختبار جيدة جداً مثل 98.0٪. لا يزال أداء خوارزمية LSTM أفضل.

```
validation accuracy score:
96.0validation precision:
96.299validation recall:
96.0validation f1 score:
95.995validation confusion matrix:
[[93  8]
 [ 0 99]]validation classification report:
              precision    recall  f1-score   support           0
1.00      0.92      0.96       101
           1       0.93      1.00      0.96        99    accuracy
0.96       200
   macro avg       0.96      0.96      0.96       200
weighted avg       0.96      0.96      0.96       200test accuracy
score:
98.0test precision:
98.078test recall:
98.0test f1 score:
98.0test confusion matrix:
[[98  4]
 [ 0 98]]test classification report:
              precision    recall  f1-score   support           0
1.00      0.96      0.98       102
           1       0.96      1.00      0.98        98    accuracy
0.98       200
   macro avg       0.98      0.98      0.98       200
weighted avg       0.98      0.98      0.98       200
```

يوضح الشكلان 5 و 6 مخططات خطأ ودقة التدريب / التحقق من الصحة نموذج LSTM. كما نرى، يمكن أن يضمن اختيار عدد الفترات بين 30 و 40 أداءً جيداً لنموذج LSTM.



الشكل 5. مخطط خطأ التدريب / التحقق من الصحة لنموذج LSTM.



الشكل 6. مخطط دقة التدريب / التحقق من الصحة لنموذج LSTM.

تم الحصول على النتائج أدناه باستخدام نموذج مصنف Multinomial Naive Bayes MultinomialNB(). حسنًا، كما ترى، لا يعمل هذا المصنف جيداً مع مجموعة البيانات الجينومية المحددة هذه. هذا مثال عملي جيد جداً لإثبات أن كل مجموعة بيانات جينية فريدة من نوعها وأن أفضل الممارسات لأي علماء بيانات هي تطبيق جميع خوارزميات التعلم الآلي التقليدية والحديثة لكل منها. يعتقد العديد من علماء البيانات أن التعلم العميق يمكنه فعل كل شيءفي التعلم الآلي لهم اليوم. من السهل جداً إثبات أنه حتى خوارزميات Random Forest و Extreme Gradient Boosting قد حققت أداءً أفضل في كثير من الحالات مقارنةً بخوارزميات التعلم العميق بمافي ذلك CNN و LSTM. لذلك، قم دائمًا بتطبيقها جميعًا على كل مجموعة بيانات جينية محددة واستخدم المجموعة ذات المقاييس الأفضل أداء. أتفهم أن هذا قد يستغرق بعض الوقت ولكنه شرط ضروري للغاية للحصول على أفضل نموذج ممكن.

```
validation accuracy score:
80.5validation precision:
86.011validation recall:
80.5validation f1 score:
79.772validation confusion matrix:
[[62 39]
 [ 0 99]]validation classification report:
              precision    recall  f1-score   support           0
1.00      0.61      0.76       101
           1       0.72      1.00      0.84        99     accuracy
0.81       200
   macro avg       0.86      0.81      0.80       200
weighted avg       0.86      0.81      0.80       200test accuracy
score:
79.5test precision:
85.547test recall:
79.5test f1 score:
78.695test confusion matrix:
```

```
[[61 41]
 [ 0 98]]test classification report:
              precision    recall  f1-score   support           0
1.00      0.60      0.75       102
          1       0.71      1.00      0.83        98    accuracy
0.80       200
   macro avg       0.85      0.80      0.79       200
weighted avg       0.86      0.80      0.79       200
```

25- الاستنتاجات

- تم تطوير مكتبة PyDNA المخصصةفي Python لمعالجة السلاسل النصية string لتسلسل RNA / DNA / البروتين ومجموعات البيانات الجينومية لتصنيف التعلم الآلي.
- لزيادة أداء معالجة السلاسل النصية تسلسلات الحمض النووي الكبيرة، يوصى باستخدام NumPy ndarrys حيثما أمكن ذلك.
- تم تقديم وتحليل السلسلة النصية لترميز تسلسل الحمض النووي الرئيسية التالية: ترميز التسميات Label Encoding، والترميز الواحد الساخن One-hot Encoding، وعد K-mer Counting K-mer.
- تم تنفيذ خوارزمية حقيبة الكلمات bag of words من معالجة اللغة الطبيعية لمعالجة السلاسل النصية لتسلسل الحمض النووي.
- يمكن أن يحدد التوحيد uniformityفي طول تسلسل الحمض النووي عبر مجموعة البيانات خوارزمية التعلم الآلي المناسبة لاستخدامها.
- توفر نماذج مصنف نايف متعدد الحدود Multinomial Naive وبيربيسترون متعدده الطبقات Multi-layer Perceptron أكثر من 97 ٪ من درجة دقة التصنيف مع مجموعة بيانات DNA متعددة الفئات ولا يوجد طول سلسلة موحد.
- يمكن أن يوفر نموذج الشبكات العصبية التلافيفية CNN 97 ٪ من دقة التصنيف مع طول سلسلة تسلسل الحمض النووي الموحد.
- يوفر نموذج الذاكرة طويلة المدى قصيرة المدى LSTM أفضل نتيجة بنسبة 99.5٪ من درجة دقة التصنيف مع طول سلاسل تسلسل الحمض النووي الموحد.
- يوصى بشدة بتطبيق جميع خوارزميات تصنيف التعلم الآلي التقليدية والحديثة على أي مجموعة بيانات جينية ومعرفة النموذج الذي يوفر أفضل نتائج التنبؤ.

المصدر:

https://medium.com/mlearning-ai/apply-machine-learning-algorithms-for-genomics-data-classification-132972933723

2) بناء نماذج تجميع تعلم الآلة للتعبير الجيني لبيانات RNA-Seq
Building Machine Learning Clustering Models for Gene Expression RNA-Seq Data

1. المقدمة

التجميع (التكتل) Clustering هو مجموعة من التقنيات المستخدمة لتقسيم البيانات إلى مجموعات groups أو تكتلات clusters. يتم تعريف التجميع بشكل فضفاض على أنها مجموعات من كائنات البيانات التي تشبه إلى حد كبير كائنات أخرى في مجموعتها أكثر من كائنات البيانات في مجموعات أخرى ("K-Means Clustering in Python: A Practical Guide"). الهدف الأساسي من التجميع هو تجميع البيانات في مجموعات بناءً على التشابه similarity أو الكثافة density أو الفواصل الزمنية intervals أو مقاييس التوزيع الإحصائي particular statistical distribution measures الخاصة لفضاء البيانات data space ("مناهج التجميع القائمة على التعلم العميق للمعلوماتية الحيوية"). بشكل عام، يساعد التجميع في تحليل البيانات غير المهيكلة وعالية الأبعاد في شكل تسلسلات وتعبيرات ونصوص وصور.

2. استخدام خوارزميات التجميع في المعلوماتية الحيوية

تحتوي بيانات التعبير الجيني Gene expression data على بيانات المصفوفة الدقيقة لـ DNA (DNA microarray data) وبيانات تسلسل RNA (RNA-seq data). يساعد تحليل بيانات المصفوفات الدقيقة على توضيح الآليات البيولوجية ودفع الأدوية نحو مستقبل أكثر قابلية للتنبؤ. مقارنةً بتقنية المصفوفة الدقيقة القائمة على التهجين hybridization-based microarray technology، فإن RNA-seq لديها نطاق أكبر من مستويات التعبير، ويتم الكشف عن المزيد من المعلومات ("إستراتيجية اختيار ميزة فعالة تعتمد على تقنية آلة المتجهات الداعمة المتعددة مع بيانات التعبير الجيني").

أحدثت المصفوفات الدقيقة للحمض النووي DNA microarrays ثورة في نهج تنميط التعبير الجيني. مقارنة بالطرق السابقة، تتمتع المصفوفات الدقيقة للحمض النووي بإنتاجية عالية جداً وأقل تعقيداً. تم تطوير المصفوفات الدقيقة كأسلوب لرسم خرائط الحمض النووي على نطاق واسع وتسلسله. ومع ذلك، فإن تغيير سطح الدعم من غشاء مسامي porous membrane إلى سطح صلب solid surface أتاح تحسينات كبيرة عن طريق زيادة حركية التفاعل وتقليل ضوضاء الخلفية. تستمر تقنية المصفوفات الدقيقة للحمض النووي في التطور ("المصفوفات الدقيقة للحمض النووي من الفصل السابع في الاكتشاف البيولوجي ورعاية المرضى").

بعض تطبيقات التجميع clusteringفي المعلوماتية الحيوية Bioinformatics يمكن أن تكون: مجموعات محددة من المرضى الذين يستجيبون بشكل مختلف للعلاجات الطبية لتحديد المرض؛ تكشف عن مجموعات من الجينات ذات الصلة وظيفيًا حيث تشترك الجينات ذات المسافة الصغيرة في نفس أنماط التعبير وقد تكون متشابهة. يمكن أن يكون تصور البيانات البيولوجية عالية الأبعاد والواسعة النطاقفي مجملها غير المنظم وتفسيرها وتحليلها أمرًا محيرًا ما لم يتم تنظيم البياناتفي مجموعات. مثال آخر هو تجميع الجينات أو الصور الطبية الحيوية من خلال تعلم الأنماط المخفية من مجموعة بيانات غير مسماة unlabeled dataset ("إستراتيجية اختيار ميزة فعالة تعتمد على تقنية آلة المتجهات الداعمة المتعددة مع بيانات التعبير الجيني").

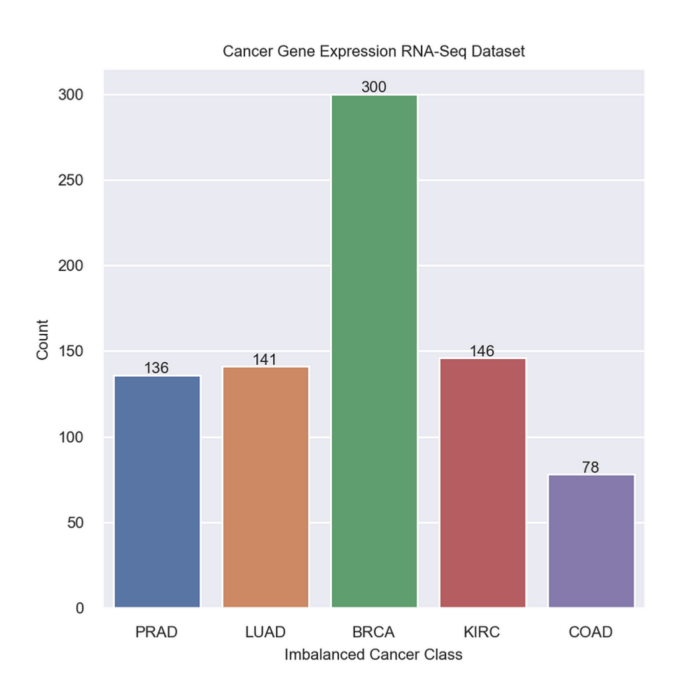
أصبح RNA-Seq البيانات الرئيسية لقياسات التعبير الجيني ويقدم العديد من المزايا على المصفوفات الدقيقة. علاوة على ذلك، أصبحت التجارب أحادية الخلية مشروعًا بحثيًا أساسيًا للمعلوماتية الحيوية، حيث تعد خوارزميات التجميع جزءًا مهمًا من تنفيذ مشاريع التعلم الآلي Machine Learning (ML).

في ورقة التعلم الآلي هذه، سيتم تطبيق خوارزمية التجميع K-Means لبناء نموذج متوقع لمجموعة بيانات تعبير الجيني لـ RNA-Seq. تعد خوارزمية التجميع هذه تقنية تعلم غير خاضعة للإشراف Unsupervised Learning تُستخدم لتحديد مجموعات كائنات البيانات متعددة الأبعادفي مجموعة بيانات. تعد خوارزمية التجميع K-Means جزءًا من مكتبة إطار عمل scikit-Learning ML الشهيرة.

3. مجموعة بيانات RNA-Seq للتعبير الجيني للسرطان

يمكن تنزيل مجموعة بيانات RNA-Seq للتعبير الجيني للسرطان من مستودع التعلم الآلي UCI. هذه المجموعة من البيانات هي جزء من مجموعة بيانات أطلس -RNA-Seq (HiSeq) Pan Cancer Atlas. هو استخراج عشوائي للتعبيرات الجينية للمرضى الذين يعانون من أنواع مختلفة من الأورام بمافي ذلك سرطان الثدي الغازي Breast invasive carcinoma (BRCA)وسرطان الخلايا الكلوية الصافية Kidney renal clear cell carcinoma ، (COAD)ـ سرطان القولون الغدي Colon adenocarcinoma،(COAD) وسرطان الغدة الرئوية Lung adenocarcinoma (LUAD) وسرطان البروستاتا Prostate adenocarcinoma (PRAD). يحتوي الملف الذي تم تنزيله، TCGA-PANCAN-HiSeq-80120531.tar.gz، على ملفين CSV. يمثل ملف data.csv الميزة Xفي التعلم الآلي ويمثل ملف label.csv بيانات الهدف y (التسمية label). تحتوي مجموعة البيانات على قيم RNA-Seq للتعبير الجيني من 20531 جينًا إلى 881 عينة.

تم توفير تحليل الفئة غير المتوازن imbalanced class analysis من مجموعة البيانات هذهفي ورقة "تصنيف التعبير الجيني RNA-Seq باستخدام خوارزميات التعلم الآلي". يحتوي المخطط الشريطي bar chart الموضح أدناه على فئة غير متوازنة لمجموعة البيانات الجينومية المختارة.

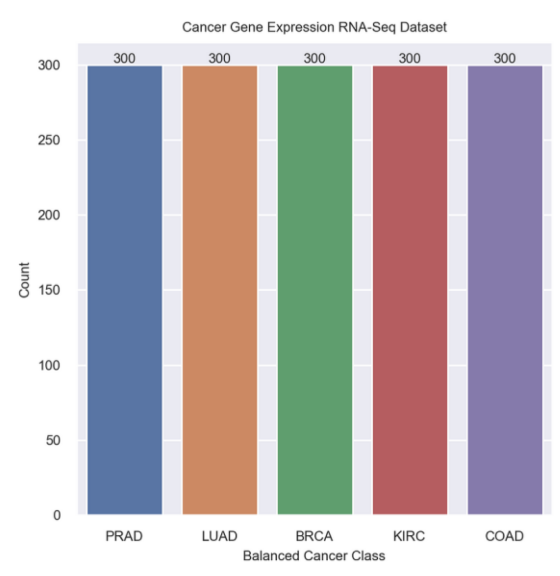


بعد تطبيق خوارزمية تقنية الإفراطفي أخذ عينات الأقليات الاصطناعية Synthetic Minority Oversampling Technique (SMOTE)، تمت موازنة جميع الفئات بـ 300 صف لكل منها كما ترونفي المخطط الشريطي الأخير أدناه. باستخدام مكتبة PyDNA، سيتم تحويل الميزة X والتسمية y باستخدام الدالة الموجودة أدناه balance_class_smote().

```
def balance_class_smote(X, y):
    """X and Y smote over sampling
    args:
        X feature
        y label
    returns:
        smote X feature
        smote y label
    """
    try:
        smote_over_sampling = SMOTE(random_state=50, n_jobs=-1)
        X, y = smote_over_sampling.fit_resample(X, y)
    except:
        print(PyDNA.get_exception_info())
        if PyDNA._app_is_log: PyDNA.write_log_file("error",
```

```
PyDNA.get_exception_info())
    return X, y

# calling function
X, y = PyDNA.balance_class_smote(X, y)
```

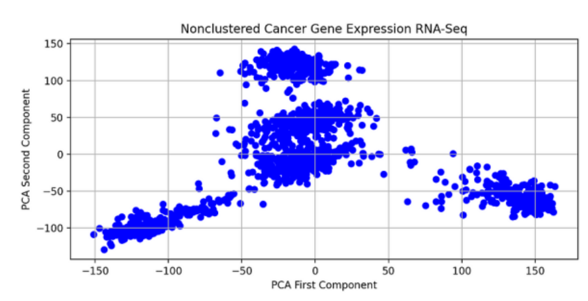


تم استخدام تحليل المكونات الرئيسية (PCA) Principal Component Analysis لتقليل عدد ميزات X (الأعمدة) إلى 7ـ10 من 20531. يمكن تصور مجموعة البيانات غير المجمعة non- clustered dataset باستخدام المكونين الأول والثاني لـ PCA. يوجد أدناه الدالة pca_x_reduction() ومخطط التصور.

```
pca = PCA(n_components=config.PCA_N_COMPONENTS_COUNT, random_state=50)
def pca_x_reduction(pca, X):
    """ X feature reduction
    args:
        pca class object
        X feature
    returns:
        X new feature
        PCA explained variance
        cumulative sum of eigen values
    """
    try:
        X = pca.fit_transform(X)
        pca_explained_variance = pca.explained_variance_ratio_
        cumulative_sum_eigenvalues = np.cumsum(pca_explained_variance)
    except:
        print(PyDNA.get_exception_info())
        if PyDNA._app_is_log: PyDNA.write_log_file("error",
PyDNA.get_exception_info())
```

```
    return X, pca_explained_variance, cumulative_sum_eigenvalues

# calling function
X = PyDNA.pca_x_reduction(pca_component_number, X)
```



4. مقاييس تقييم تجميع K-Means

على عكس التعلم الخاضع للإشراف الآلي Machine Supervised Learning حيث لدينا الحقيقة الأساسية لتقييم أداء النموذج، لا يحتوي تحليل المجموعات على مقياس تقييم evaluation metric قوي يمكننا استخدامه لتقييم نتائج خوارزميات التجميع المختلفة. علاوة على ذلك، نظرًا لأن K-Means تتطلب عدد المجموعات k كمدخلات ولا تتعلمها من البيانات، فلا توجد إجابة صحيحة من حيث k يجب أن تكون لديناقي أي مشكلة.في بعض الأحيان قد تساعد المعرفة والحدس بالمجال ولكن هذا ليس هو الحال عادةً.في منهجية التنبؤ التكتلي cluster-predict methodology، يمكننا تقييم مدى جودة أداء النماذج بناءً على مجموعات k (k clusters) المختلفة حيث يتم استخدام المجموعاتفي النمذجة النهائية. يمكنني أن أوصيت الجميع بقراءة وفهم الورقة البحثية "تجميع K-mean: الخوارزمية، والتطبيقات ، وطرق التقييم ، والعيوب".

5. تحديد عدد المجموعات باستخدام طريقة الكوع

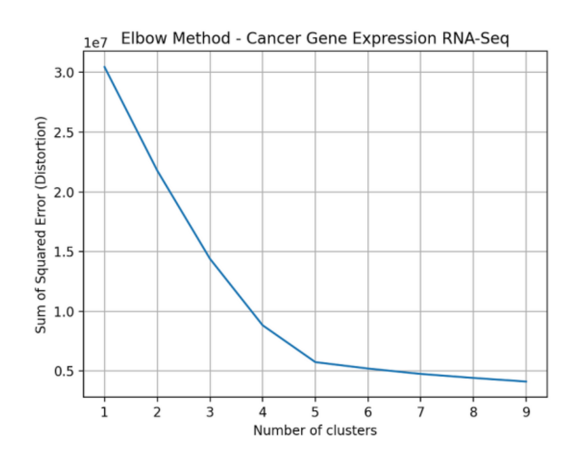
طريقة الكوع Elbow Method هي الطريقة الأكثر شيوعًا لتحديد العدد الأمثل للعناقيد (المجموعات) clusters. تستخدم هذه الطريقة معلمة مجموع الخطأ التربيعي Sum of Squared Error (SSE) بين نقاط البيانات data points والنقاط الوسطى للمجموعات clusters' centroids المخصصة لها. يجب تحديد قيمة المجموعة kفي المكان الذي يبدأ فيه SSEفي التسوية وتشكيل كوع. يوجد أدناه كود لحساب SSE ومخطط طريقة الكوع.

```
def calculate_kmeans_sse(X, max_elbow_number):
    """ calculate sum of squared error (sse)
    args:
        X feature
        max of elbow number
    returns:
        SSE
    """
    try:
```

```
        sse = []
        for i in range(1, max_elbow_number):
            kmeans = KMeans(n_clusters=i, random_state=50)
            kmeans.fit(X)
            sse.append(kmeans.inertia_)
    except:
        print(PyDNA.get_exception_info())
        if PyDNA._app_is_log: PyDNA.write_log_file("error",
PyDNA.get_exception_info())
    return sse

def elbow_method_plot(sum_squared_error, max_elbow_number):
    """elbow method clustering plot
    args:
       sum squared error
       max of elbow number
    return:
       None
    """
    try:
        plt.grid(True)
        plt.plot(range(1, max_elbow_number), sum_squared_error)
        plt.title("Elbow Method - Cancer Gene Expression RNA-Seq")
        plt.xlabel("Number of clusters")
        plt.ylabel("Sum of Squared Error (Distortion)")
        plt.show()
    except:
        print(PyDNA.get_exception_info())
        if PyDNA._app_is_log: PyDNA.write_log_file("error",
PyDNA.get_exception_info())

# calling functions
max_elbow_number = config.MAX_ELBOW_NUMBER
sum_squared_error = PyDNA.calculate_kmeans_sse(X, max_elbow_number)
PyDNA.elbow_method_plot(sum_squared_error, max_elbow_number)
```



كما نرى، يجب أن يكون العدد الصحيح من المجموعات خمسة. كل واحد منهم يمثل فئة السرطان. هناك طرق أخرى متاحة لتحديد العدد الأمثل للمجموعات بما في ذلك في المقالة "5 طرق لتقرير عدد المجموعات في نموذج الكتلة".

6. استخدام مكتبة Kneed لتحديد عدد المجموعات

تُستخدم مكتبة kneed للكشف عن نقطة الركبة knee-point detection في Python. بالنظر إلى مجموعة من قيم x و y، سيعيد kneed نقطة الركبة للدالة. نقطة الركبة knee-point هي نقطة الانحناء الأقصى point of maximum curvature. إذا كانت طريقة الكوع صعبة للغاية لتحديد عدد المجموعات، فإنني أوصي باستخدام هذه المكتبة للتحقق من العدد الأمثل الصحيح للمجموعات optimal number of clusters وتحديده. يوجد أدناه كود دالة get_kneed_elbow_point() الذي يحسب خمس مجموعات. كما ترون نتيجة مكتبة kneed المطبقة، تحقق من صحة مستخدم الاختيار الصحيح باستخدام طريقة الكوع.

```
def get_kneed_elbow_point(x_coordinate , y_coordinate ):
    """determine the kneed elbow point
    args:
        x coordinate
        y coordinate
    returns:
        kneed elbow point
    """
    try:
        kneed_locator = KneeLocator(x=x_coordinate, y=y_coordinate,
curve="convex", direction="decreasing")
        kneed_locator_elbow = kneed_locator.elbow
    except:
        print(PyDNA.get_exception_info())
        if PyDNA._app_is_log: PyDNA.write_log_file("error",
PyDNA.get_exception_info())
    return kneed_locator_elbow

# calling functions
max_elbow_number = config.MAX_ELBOW_NUMBER
sum_squared_error = PyDNA.calculate_kmeans_sse(X, max_elbow_number)
kneed_locator_elbow = PyDNA.get_kneed_elbow_point(range(1,
max_elbow_number), sum_squared_error)
print(kneed_locator_elbow)

Result: 5
```

7. تطبيق نموذج التجميع K-Means

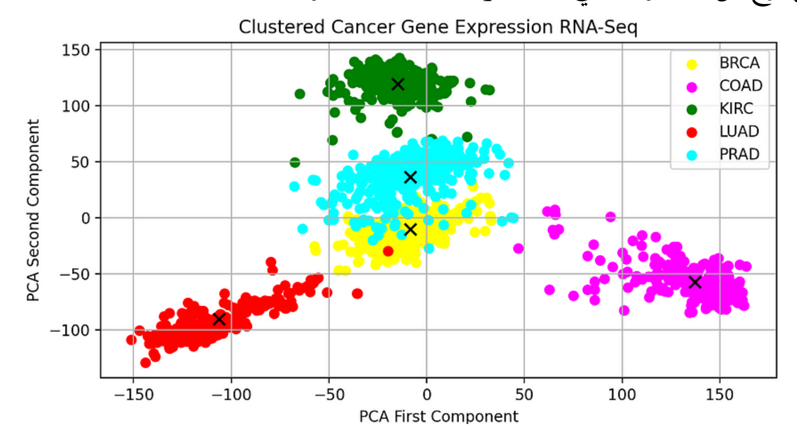
الآن بعد أن علمنا أن عدد المجموعات هو خمسة، يمكن تطبيق نموذج K-Means كما هو موضح أدناه.

```
number_clusters = config.NUMBER_CLUSTERS
k_means = PyDNA.k_means_model(number_clusters)
y_predicted = PyDNA.k_means_predict(X)
cluster_centroids =
PyDNA.k_means_cluster_centroids(kmeans.cluster_centers_)
print(cluster_centroids)
```

فيما يلي النقط الوسطى للمجموعة cluster centroids لمجموعة البيانات المحددة.

```
[[  -8.45801335  -10.14582353   76.74888149  -57.75948388    4.03079771
    16.34636755                                               -5.99093303]
 [ 137.31363915  -57.41108246  -33.51073392    4.60164944    5.01152715
     0.58311618                                                1.11230399]
 [ -14.69712212  119.25252845  -58.71502055  -22.30000913   -1.16636062
     3.59466095                                                5.54235961]
 [-106.38275644  -90.37657514  -41.62782788    2.08651512   -9.49929942
   -11.24228754                                               -1.96960122]
 [  -8.43637589   36.25531939   54.76312094   72.61165042    1.44742792
    -9.41736093    1.29875299]]
```

باستخدام نموذج K-Means والتسميات المتوقعة predicted labels، يمكننا رسم المجموعات الخمس لكل نوع من للتعبير الجيني لبيانات RNA-seq للسرطان.



8. مقياس مؤشر راند المعدل لـ K-Means

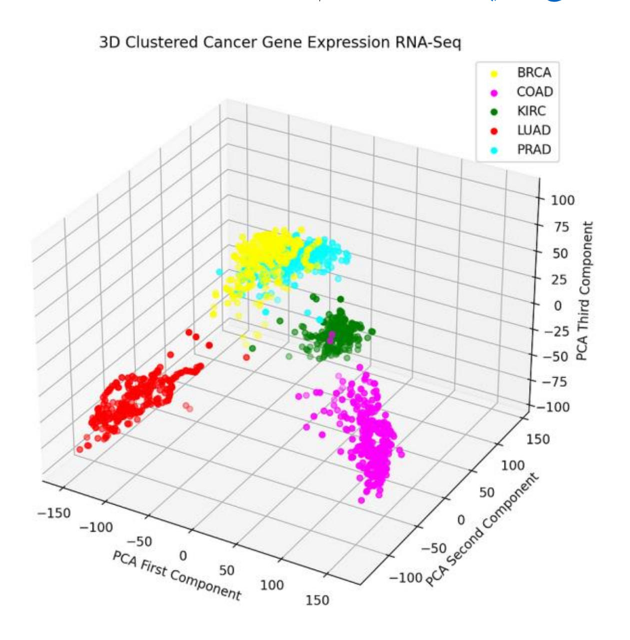
المقياس الشائع للتحقق من أداء نموذج K-Means يسمى مؤشر Rand المعدل Adjusted Rand Index (ARI). على عكس معامل Silhouette (Silhouette Coefficient)، يستخدم ARI تعيينات مجموعة حقيقية لقياس التشابه بين التسميات الحقيقية والمتوقعة. تتراوح قيم إخراج ARI بين ــ1.0 و 1.0. تشير الدرجة القريبة من 0.0 إلى تعيينات عشوائية، وتشير الدرجة القريبة من 1.0 إلى مجموعات مسماة تمامًا. يتم تحديد ARI أدناه بمقدار ــ0.5 للتجمعات المتنافرة بشكل خاص. يحسب سطر الكود أدناه قيمة ARI باستخدام الدالة alculate_adjusted_rand_score(). يمثل الرقم 0.98 أداءً جيدًا جدًا لنموذج K-Means المحدد لدينا.

```
def calculate_adjusted_rand_score(y, y_predicted):
    """ calculate the adjusted rand score
    args:
        y label
        y predicted label
    returns:
        adjusted rand score
    """
    try:
        adjusted_rand_index = adjusted_rand_score(y, y_predicted)
        adjusted_rand_index =
float("{0:0.2f}".format(adjusted_rand_index))
    except:
        print(PyDNA.get_exception_info())
        if PyDNA._app_is_log: PyDNA.write_log_file("error",
PyDNA.get_exception_info())
    return adjusted_rand_index

# calling function
adjusted_rand_index = PyDNA.calculate_adjusted_rand_score(y,
y_predicted)
print(adjusted_rand_index)

Result: 0.98
```

يمكن إنشاء مخطط مجمع ثلاثي الأبعاد باستخدام مكتبة matplotlib.



9. تحليل صورة Silhouette لتجميع K-Means

يستخدم تحليل Silhouette (Silhouette Analysis) لدراسة المسافة الفاصلة بين العناقيد الناتجة. يعرض مخطط صورة Silhouette مقياسًا لمدى قرب كل نقطةفي مجموعة واحدة من النقاط الموجودة في المجموعات المجاورة، وبالتالي يوفر طريقة لتقييم المعلمات مثل عدد المجموعات بصريًا. هذا المقياس له مدى [-1, 1].

للمساعدةفي تحديد العدد الأمثل للمجموعات، يوفر تحليل صورة Silhouette هذا مقياسين رئيسيين. silhouette_samples() ـ حساب معامل Silhouette لكل عينة. أفضل قيمة هي 1 وأسوأ قيمة هي ـ1. تشير القيم القريبة من 0 إلى مجموعات متداخلة. silhouette_score() ـ حساب متوسط معامل صورة Silhouette لجميع العينات. ترجع هذه الدالة متوسط معامل صورة Silhouette على جميع العينات. للحصول على قيم كل عينة، استخدم silhouette_samples(). أفضل قيمة هي 1 وأسوأ قيمة هي ـ1. تشير القيم القريبة من 0 إلى مجموعات متداخلة. تشير القيم السالبة بشكل عام إلى أنه تم تعيين عينة إلى المجموعة الخاطئة، حيث أن المجموعة المختلفة أكثر تشابهًا. هذا هو الكود لحساب قيمة نتيجة صورة Silhouette.

```
def calculate_print_silhouette_score(X):
    """calculate silhouette score for each cluster number
    args:
        X feature
    returns
        None
    """
    try:
        range_number_clusters = [2, 3, 4, 5]
        for number_cluster in range_number_clusters:
            print(number_cluster)
            kmeans = KMeans(n_clusters=number_cluster,
random_state=50)
            y_cluster_labels = kmeans.fit_predict(X)
            silhouette_score_avg = silhouette_score(X,
y_cluster_labels)
            print(silhouette_score_avg)
    except:
        print(PyDNA.get_exception_info())
        if PyDNA._app_is_log: PyDNA.write_log_file("error",
PyDNA.get_exception_info())

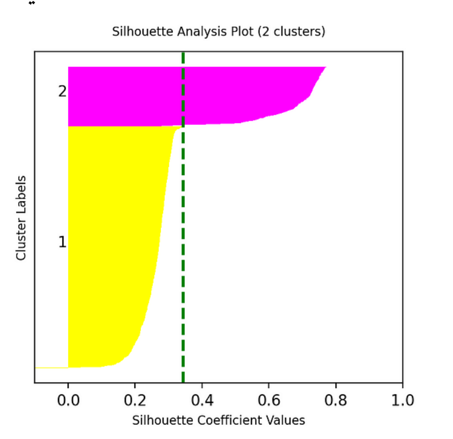
# calling function
PyDNA.calculate_print_silhouette_score(X)
```

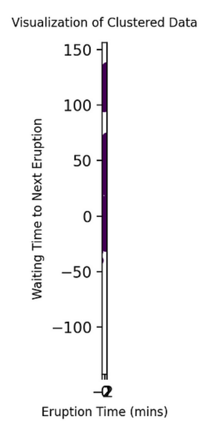
بالنسبة لمجموعة البيانات الخاصة بنا، يعرض الجدول أدناه قيم درجات صورة Silhouette المحسوبة لعدد محدد من المجموعات.

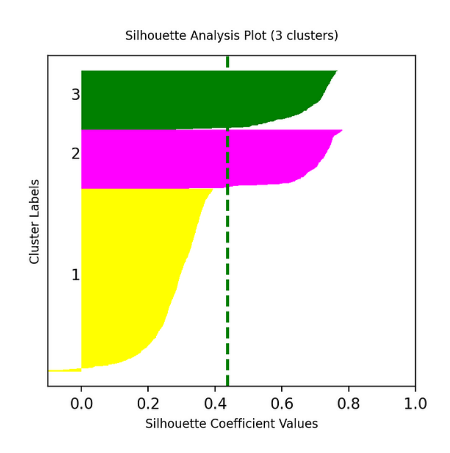
Number of Clusters	Silhouette Score
2	0.34
3	0.43
4	0.52
5	0.57

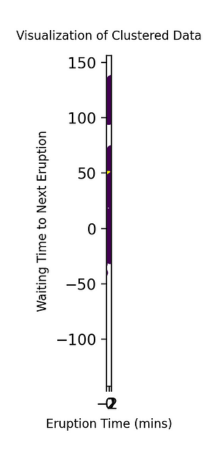
كما يمكننا أن نرى أن أعلى قيمة لدرجة صورة Silhouette هي 0.57 لخمس مجموعات. لذلك، تم التحقق من العدد الأمثل للمجموعات لمجموعة البيانات الخاصة بنا على أنه خمسة، كما ينبغي أن يكون من طرق الكوع ونتائج حساب مكتبة Kneed.

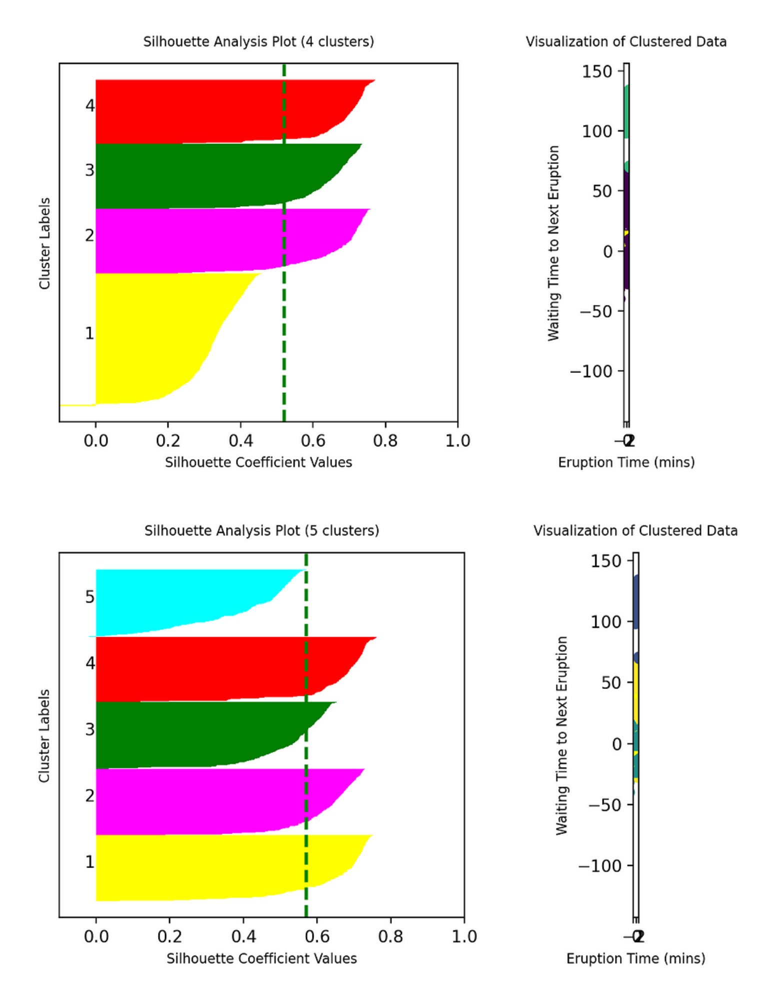
يظهر مخطط الصورة Silhouette أدناه. يعطي سمك هذه المخطط مؤشرا على حجم كل مجموعة. إنه يثبت أنه بالنسبة لخمس مجموعات، أعلى قيمة لدرجة صورة Silhouette هي 0.57.











10. الاستنتاجات

1. أثبتت خوارزمية التجميع K-Means أنها خيار جيد للتعبير الجيني متعدد الأبعاد لمجموعات بيانات RNA-Seq.
2. توفر طريقة الكوع حلاً رسوميًا بسيطًا لتحديد العدد الأمثل للمجموعات.
3. يمكن لمكتبة Need أن تحدد وتتحقق من العدد الأمثل للمجموعات للتعبير الجيني متعدد الأبعاد لمجموعات بيانات RNA-Seq.
4. تعد قيمة مؤشر Rand المعدل مقياسًا ممتازًا للتحقق من أداء نموذج K-Means.
5. يوفر تحليل صورة Silhouette حلاً جيدًا لتحديد العدد الأمثل للمجموعات باستخدام حسابات عينات صور Silhouette ومقاييس النتيجة score metrics.

المصدر:

https://ernest-bonat.medium.com/building-machine-learning-clustering-models-for-gene-expression-rna-seq-data-d0e5af10416d

3) تصنيف التعبير الجيني لـ RNA-Seq باستخدام خوارزميات التعلم الآلي
RNA-Seq Gene Expression Classification Using Machine Learning Algorithms

1. نظرة عامة

من موقع المعهد القومي لبحوث الجينوم البشري National Human Genome Research Institute (NHGRI) يمكننا أن نرى بعض أمثلة تطبيقات الذكاء الاصطناعي / التعلم الآلي في علم الجينوم.

- فحص وجوه الأشخاص باستخدام برامج الذكاء الاصطناعي لتحليل الوجه لتحديد الاضطرابات الوراثية بدقة.
- استخدام تقنيات التعلم الآلي لتحديد النوع الأساسي للسرطان من الخزعة السائلة liquid biopsy.
- توقع كيفية تطور نوع معين من السرطان لدى المريض.
- تحديد المتغيرات الجينية genomic variants المسببة للأمراض مقارنة بالمتغيرات الحميدة benign variants باستخدام التعلم الآلي.
- استخدام التعلم العميق لتحسين وظيفة أدوات تحرير الجينات مثل كريسبر CRISPR.

دعونا نعرف التعبير الجيني gene expression. التعبير الجيني هو العملية التي يتم من خلالها استخدام المعلومات المشفرة في الجين إما لصنع جزيئات RNA التي ترمز للبروتينات أو لصنع جزيئات RNA غير مشفرة تخدم وظائف أخرى. يعمل التعبير الجيني بمثابة "مفتاح تشغيل / إيقاف" للتحكم في متى وأين يتم تصنيع جزيئات وبروتينات الحمض النووي الريبي (RNA) و"التحكم في الحجم volume control" لتحديد مقدار هذه المنتجات التي يتم تصنيعها. يتم تنظيم عملية التعبير الجيني بعناية، وتتغير بشكل كبير في ظل ظروف مختلفة. تعمل منتجات RNA والبروتين للعديد من الجينات على تنظيم التعبير عن الجينات الأخرى.

ستغطي المقالة هذه كيفية تطبيق خوارزميات التعلم الآلي للتصنيف على مجموعة بيانات التعبير الجيني RNA-Seq. بشكل عام، تعد هذه الأنواع من مجموعات البيانات متعددة الأبعاد وتحتوي على آلاف الأعمدة غير المسماة unlabeled columns. بسبب هذا التطبيق المحدد للتعلم الآلي ضروري لتطوير نماذج إنتاج عالية الأداء. سيتم توفير خطوات سير عمل التعلم الآلي المطلوبة باستخدام نماذج التصنيف الأكثر شيوعًا اليوم. ورقة البحث هذه هي استمرار لتطوير نماذج التعلم الآلي ونشرها لتحليل مجموعات البيانات الجينومية. أدناه هو الأحدث:

- "تطبيق خوارزميات التعلم الآلي لتصنيف بيانات الجينوم ". إرنست بونات، دكتوراه، بيش رياماجي، ماجستير. 03 فبراير 2021.
- "نشر الويب لنماذج التعلم الآلي لعلم الجينوم باستخدام Flask Web Framework". إرنست بونات، دكتوراه ، 18 يونيو 2022.
- "بناء نماذج مجموعات تعلم الآلة لبيانات RNA-Seq للتعبير الجيني". إرنست بونات، دكتوراه ، 28 ديسمبر 2022.

2. تحميل سريع لمجموعة بيانات التعبير الجيني RNA-Seq

في الورقة هذه، سيتم استخدام مجموعة بيانات RNA-Seq الخاصة بسرطان التعبير الجيني. هذه المجموعة من البيانات هي جزء من مجموعة بيانات أطلس RNA-Seq (HiSeq) Pan-Cancer Atlas، وهي عبارة عن استخراج عشوائي للتعبيرات الجينية للمرضى الذين يعانون من أنواع مختلفة من الأورام بمافي ذلك BRCA ـ سرطان الثدي الغازي، KIRC ـ سرطان الخلايا الكلوية الصافية، COAD ـ سرطان القولون الغدي، LUAD ـ سرطان الغدة الرئوية وPRAD ـ سرطان البروستاتا الغدي. يحتوي الملف الذي تم تنزيله TCGA-PANCAN-HiSeq-80120531.tar.gz على ملفين CSV. ملف data.csv الذي يمثل الميزة X في التعلم الآلي وملف lable.csv الذي تمثل بيانات (التسمية) الهدف y. يوجد أدناه معلومات إطار بيانات مكتبة الباندا لكل منهم. Pandas هي أداة تحليل ومعالجة بيانات مفتوحة المصدر سريعة وقوية ومرنة وسهلة الاستخدام، مبنية على قمة لغة برمجة Python.

لميزة X (ملف data.csv).

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 801 entries, 0 to 800
Columns: 20531 entries, gene_0 to gene_20530
dtypes: float64(20531)
memory usage: 125.5 MB
None
```

أثناء تطوير نموذج التعلم الآلي واختباراته، ستحتاج ملفات CSV هذه إلى التحميل عدة مرات في برنامج Python IDE. بشكل عام، تبلغ المدة الزمنية لتحميل هذه الملفات حوالي 16ـ18 ثانية (وفقاً لجهاز الكمبيوتر المحمول الخاص بي). سيؤدي هذا إلى زيادة كبيرة في وقت التطوير للتنفيذ النهائي للنموذج الإجمالي. السؤال هو: كيف يمكن تسريع عملية تحميل البيانات؟ تتمثل إحدى الطرق السهلة في إجراء تسلسل serialize للكائنات X و y مرة واحدة ثم إلغاء تسلسلها deserialize عدة مرات. في المرة التالية التي تقوم فيها بتشغيل ملف Python، ستحتاج إلى إلغاء تسلسل X و y فقط حسب الحاجة. يمكن لتقنية تحميل ملفات Python CSV أن تقلل من وقت التطوير حتى 1ـ2 ثانية لكل تنفيذ ـ فكرة ممتازة بنتيجة جيدة! يمكن إجراء تسلسل الكائن وإلغاء التسلسل في Python عن طريق

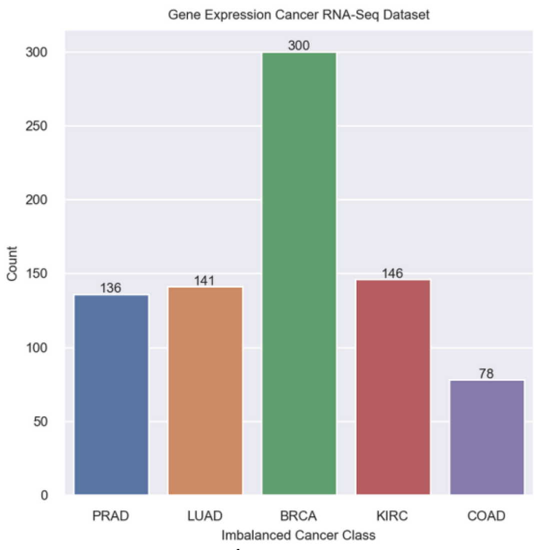
استيراد وحدات pickle أو joblib. أود أن أوصي كل مطور تعلم آلي بتنفيذ تقنية تحميل البيانات متعددة الأبعاد multi-dimensional data loading السريعة إذا لزم الأمر.

إليك كود PyDNA لتسلسل وإلغاء تسلسل كائنات الباندا X و y.

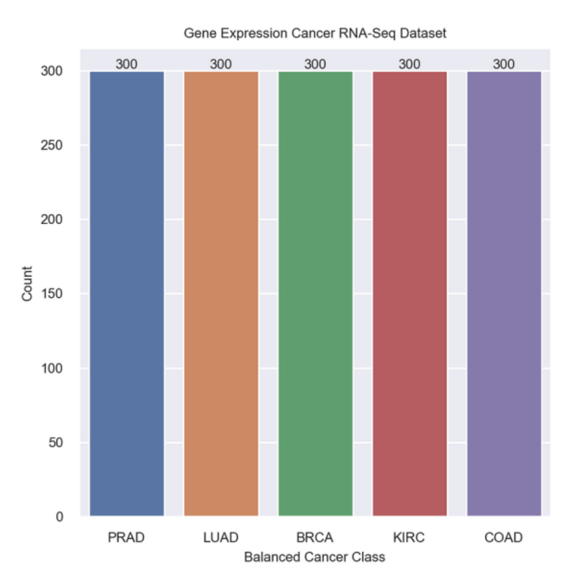
```
# X feature:rna_seq_X_path = os.path.join(rna_seq_pkl_path,
"rna_seq_X.pkl")
PyDNA.pickle_serialize_object(rna_seq_X_path, X)
X = PyDNA.pickle_deserialize_object(rna_seq_X_path)# y
target:rna_seq_y_path = os.path.join(rna_seq_pkl_path,
"rna_seq_y.pkl")
PyDNA.pickle_serialize_object(rna_seq_y_path, y)
y = PyDNA.pickle_deserialize_object(rna_seq_y_path)
```

3. تحليل الفئة غير المتوازن لمجموعة بيانات التعبير الجيني RNA-Seq

تعد الفئة غير المتوازنة imbalanced class للبيانات المستهدفةفي مجموعات بيانات الأعمال الحقيقية مشكلة شائعة جداً اليوم. هناك العديد من الأوراق المكتوبة حول مسألة التصنيف المهمة هذه المطبقة على مشاريع التعلم الآلي. على سبيل المثال، تحتوي ورقة كيفية التعامل مع البيانات غير المتوازنة بايثون على معلومات ممتازة حول حلول بايثون لموازنة الفئة غير المتوازنة المستهدفة. يحتوي المخطط الشريطي الموضح أدناه على فئة غير متوازنة لمجموعة البيانات الجينومية المختارة.



بعد تطبيق خوارزمية تقنية الإفراطفي أخذ عينات الأقليات الاصطناعية (SMOTE)، تمت موازنة جميع الفئات بـ 300 صف لكل منها كما ترونفي المخطط الشريطي الأخير أدناه.



تم تطبيق سطر بسيط من كود Python للحصول على هذا المطلب.

```
X_train, y_train = PyDNA.balance_class_smote(X_train, y_train)
```

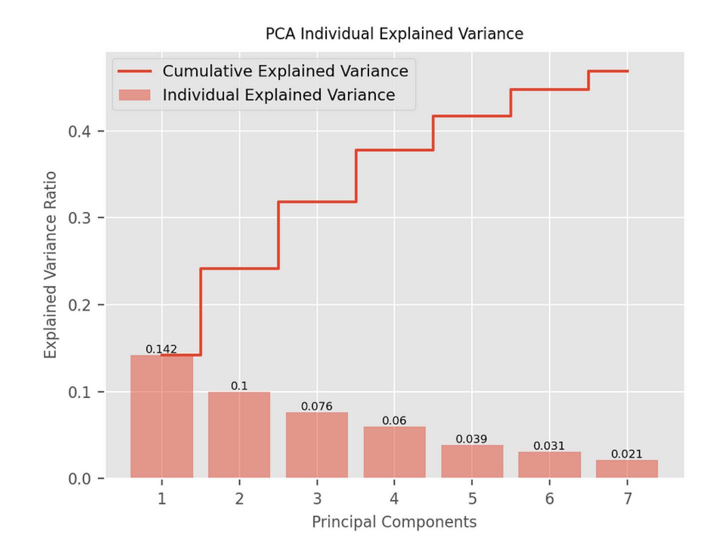
4. تطبيق تحليل المكون الرئيسي (PCA) على مجموعة بيانات التعبير الجيني لـ RNA-Seq

كما نعلم بالفعل، يحتوي إطار بيانات ميزة X على 20531 عمودًا للتعبير الجيني. سنحتاج إلى تقليل هذا العدد من الأعمدة بشكل كبير لتطبيق خوارزميات التعلم الآلي بشكل صحيح على مجموعة البيانات هذه. الأسئلة الرئيسية هنا هي كيفية تحديد عدد الأعمدة التي يجب تقليلها؟ دعونا نلقي نظرة على حل تقليل الميزات أولاً.

تقليل الميزات Feature reduction، المعروف أيضًا باسم تقليل الأبعاد dimensional reduction، هو عملية تقليل عدد الميزات في الحسابات ثقيلة الموارد دون فقدان المعلومات المهمة. تقليل عدد الميزات يعني تقليل عدد المتغيرات مما يجعل عمل الكمبيوتر أسهل وأسرع. يمكن تقسيم تقليل الميزات إلى عمليتين: اختيار الميزة feature selection واستخراج الميزة feature extraction. هناك العديد من التقنيات التي يتم من خلالها تقليل الميزات. بعض من أكثرها شيوعًا هي تحليل التمييز المعمم generalized discriminant analysis، والمشفرات التلقائية autoencoders، وعوامل المصفوفة غير السلبية non-negative matrix factorization، و A One-Stop Shop لتحليل المكونات الرئيسية PCA. PCA هي الخوارزمية الأكثر شيوعًا المستخدمة لتقليل الميزات في سير عمل مشاريع التعلم الآلي اليوم.

هناك طريقة بسيطة لتحديد الكمية الضرورية للمكونات الرئيسية عن طريق حساب نسبة التباين المفسرة الفردية. تحتوي الورقة البحثية شرح PCA مفاهيم التباين باستخدام مثال Python على شرح جيد جداً لهذه التقنية بما في ذلك كود Python أيضًا.

من المخطط الشريطي أدناه، يمكننا أن نرى أن أخذ 7ـ8 مكونات رئيسية principal components من 20531 عمودًا سيكون كافيًا لاستخدام أي خوارزميات "تعلم الآلة" للتصنيف classification. بالطبع، ستقرر قيم مقاييس التصنيف النهائية ما إذا كان هذا الاختيار صحيحًا أم لا. سيتم تقديم المزيد من الشرح حول هذا الموضوع بعد تطبيق خوارزميات "التعلم الآلي" للتصنيف.



في تجربتي، تعمل PCA بشكل جيد للغاية بالنسبة لي مع العديد من مجموعات البيانات عالية الأبعاد. إليك كود Python لحساب نسبة التباين الموضحة والمجموع التراكمي لقيم المتجهات الذاتية eigenvectors لجميع المتجهات الذاتية.

```
pca = PCA(n_components=config.PCA_N_COMPONENTS_COUNT, random_state=100)
X_train, X_valid, pca_explained_variance, cumulative_sum_eigenvalues =
PyDNA.X_train_valid_pca(pca, X_train, X_valid)
```

5. تطبيق خوارزميات تصنيف التعلم الآلي على مجموعة بيانات التعبير الجيني لـ RNA-Seq

مطلوب خطوات سير عمل التعلم الآلي الاثنتي عشرة التالية لتطبيق أي خوارزمية تصنيف تقليدية:

1. تحميل البيانات وتسلسل الكائن Data loading and object serialization.
2. إلغاء تسلسل كائن البيانات Data object deserialization.

3. تسطيح البيانات الهدف وترميزها Target data flattening and encoding.
4. البيانات مقسمة إلى تدريب training (٪80)، التحقق من الصحة validation (٪10) واختبار test (٪10)
5. توازن بيانات تدريب ميزة Balance feature training data إذا لزم الأمر.
6. تحجيم بيانات تدريب ميزة Feature training data scaling.
7. تقليل أبعاد بيانات الميزةPCA . Feature data PCA dimensional reduction
8. تطوير نموذج التصنيف وتحسينه classification model development and optimization.
9. تطبيق نموذج التصنيف Classification model fitting.
10. التنبؤ بنموذج الهدف Model target prediction.
11. حساب مقاييس التصنيف Classification metrics للتحقق من بيانات التحقق من الصحة والاختبار
12. نشر نموذج الإنتاج واداماه Production model deployment and maintenance

يوضح الجدول أدناه نتائج تطبيق اثنتي عشرة خوارزمية تصنيف كلاسيكية للتعلم الآلي لمجموعة البيانات المختارة لدينا. مقاييس إخراج التصنيف الرئيسية هي درجات دقة التحقق من الصحة والاختبار validation and test accuracy scores.

Classification Algorithm	Validation Accuracy Score, %	Test Accuracy Score, %
Logistic Regression	95.0	96.29
Decision Tree	90.0	93.82
Random Forest	97.50	97.53
Support Vector Machine	93.75	96.29
Multinomial Naive Bayes	95.0	95.62
K-Nearest Neighbors	97.50	98.71
Multi-Layer Perceptron	97.51	97.53
Gradient Boosting	97.50	93.82
Ada Boost	86.25	80.24
Extreme Gradient Boosting	97.50	95.06
Light Gradient Boosting Machine	98.75	98.76
CatBoost	96.25	97.76

الفائز هو آلة تعزيز التدرج الخفيف Light Gradient Boosting Machine (LightGBM) بنسبة 98.76٪ من درجات دقة اختبار التصنيف. LightGBM هو إطار عمل مجاني ومفتوح المصدر لتعزيز التدرج الموزع للتعلم الآلي تم تطويره في الأصل بواسطة Microsoftفي عام 2016. ويستند إلى خوارزميات شجرة القرار Decision Tree ويستخدم للترتيب ranking والتصنيف classification

ومهام التعلم الآلي الأخرى. ينصب تركيز التطوير على الأداء performance وقابلية التوسع scalability.

فيما يلي نتائج البرنامج لخوارزمية المصنف LightGBM.

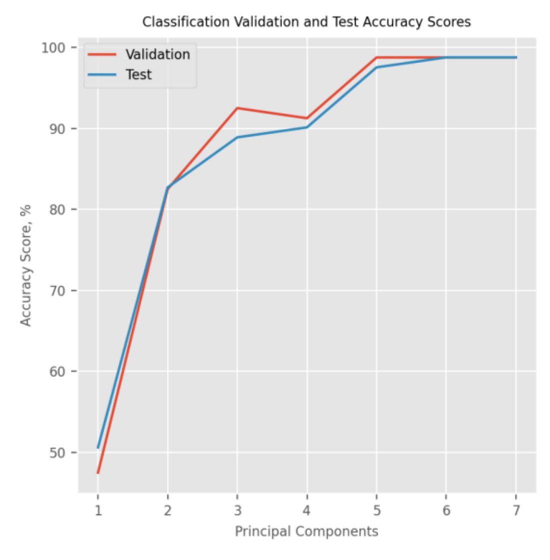
```
MODEL VALIDATIONvalid accuracy score:
98.75valid precision:
98.88valid recall:
98.75valid f1 score:
98.76valid confusion matrix:[[30  0  0  0  0]
 [ 0  8  0  0  0]
 [ 0  0 15  0  0]
 [ 0  1  0 13  0]
 [ 0  0  0  0 13]]valid classification report:       precision
recall    f1-score    support0.0       1.00      1.00      1.00
30
1.0        0.89      1.00      0.94         8
2.0        1.00      1.00      1.00        15
3.0        1.00      0.93      0.96        14
4.0        1.00      1.00      1.00        13accuracy
0.99         80
macro avg        0.98      0.99      0.98        80
weighted avg     0.99      0.99      0.99        80MODEL TESTtest
accuracy score:
98.76test precision:
98.84test recall:
98.76test f1 score:
98.76test confusion matrix:[[30  0  0  0  0]
 [ 0  8  0  0  0]
 [ 0  0 14  1  0]
 [ 0  0  0 14  0]
 [ 0  0  0  0 14]]test classification report:        precision
recall    f1-score    support0.0       1.00      1.00      1.00
30
1.0        1.00      1.00      1.00         8
2.0        1.00      0.93      0.97        15
3.0        0.93      1.00      0.97        14
4.0        1.00      1.00      1.00        14accuracy
0.99         81
macro avg         0.99      0.99      0.99        81
weighted avg      0.99      0.99      0.99        81
```

من الجيد الإشارة إلى مدى جودة عمل خوارزميات تعزيز التدرج gradient boosting algorithms مع هذا النوع من مجموعات البيانات غير المصنفة unlabeled dataset. توفر خوارزميات الغابة العشوائية Random Forest (RF) وبيرسيبترون متعدد الطبقات Multi-Layer Perceptron (MLP) نتائج جيدة جداً بنسبة دقة تبلغ 97.53٪. هذه علامة جيدة جداً لأن العديد من الشركات تستخدم RF كخوارزمية رئيسية لمشروع التعلم الآلي. أثبت RF أنه ينشئ نماذج انحدار وتصنيف

ممتازة مع العديد من مجموعات البيانات التجارية المختلفة للشركة. كانت نتيجة الاهتمام الأخرى هي خوارزمية (KNN) K-Nearest Neighbours بدرجة دقة عالية بلغت 98.71٪. هذه هي المرة الأولى التي رأيت فيها هذه النتيجة. من المنطقي تجربة KNN بمجموعات بيانات مختلفة للتعبير الجيني. في ورقة مستقبلية أود معرفة مدى جودة خوارزميات التعلم العميق مع هذا النوع من مجموعات البيانات. يمكن أن يكون السؤال الرئيسي: هل يمكن أن يكون أداء CNN أو LSTM أفضل من LigthGBM بنسبة 98.76٪ من درجة دقة التصنيف؟

6. اختيار مكونات كمية PCA باستخدام تباين درجات دقة التصنيف

تعد درجة الدقة accuracy score أحد المقاييس الرئيسية للتحقق من أداء نماذج التصنيف. من الواضح أنه بناءً على تحليل PCA وجدول نتائج خوارزميات التصنيف، فإن 7ـ8 مكونات رئيسية principal components ستؤدي المهمة. هناك طريقة أخرى، لم أرها من قبل، وهي تصور كيف تختلف درجات التحقق من الصحة والاختبار بناءً على كمية المكونات الرئيسية المختارة. يوضح الشكل أدناه كيف حصلت كلتا درجات الدقة على القيم القصوى واقتربت من ست مكونات رئيسية محددة. هذا دليل آخر على اختيارنا الصحيح المكون من 7ـ8 مكونات رئيسية.



يوجد أدناه كود Python لحساب مقاييس تصنيف التحقق من الصحة validation classification metrics المطلوبة.

```
print("MODEL VALIDATION")
accuracy_score_value, precision_value, recall_value, f1_score_value,
confusion_matrix_value, classification_report_value =
PyDNA.calculate_classification_metrics(y_val, y_predicted)
print("valid accuracy score:\n{}\n".format(accuracy_score_value))
```

```
print("valid precision:\n{}\n".format(precision_value))
print("valid recall:\n{}\n".format(recall_value))
print("valid f1 score:\n{}\n".format(f1_score_value))
print("valid confusion matrix:\n{\n".format(confusion_matrix_value))
print("valid classification report:\n{
\n".format(classification_report_value))
```

7. الاستنتاجات

1. ستؤدي كائنات pandas التي يتم تحميلها إلى التسلسل serialize وإلغاء التسلسل deserialize إلى تسريع عملية تطوير نماذج التعلم الآلي وتحسينها
2. الفئة غير المتوازنة المستهدفة Target imbalanced class هي مشكلة يجب حلها قبل تطبيق خوارزميات التعلم الآلي للتصنيف. يمكن أن يكون استخدام تقنية SMOTE مكاناً جيداً للبدء.
3. سيكون تقليل الميزات Feature reduction مطلوباً لمجموعات بيانات التعبير الجيني متعدد الأبعاد. يمكن أن يكون PCA هو الخيار الأمثل مع تحليل التباين الموضح الفردي والتصور.
4. تضمن نماذج تعزيز التدرج التقليدية للتصنيف Traditional classification gradient boosting models نتائج مقاييس جيدة مع مجموعة بيانات Pan-Cancer Atlas.
5. يمكن التحقق من صحة اختيار كمية المكون الرئيسي principal component quantity وتصوره باستخدام اختلاف درجات دقة التصنيف.
6. يوصى بشدة بتطبيق جميع خوارزميات تصنيف التعلم الآلي التقليدية والحديثة على أي مجموعة بيانات جينية ومعرفة النموذج الذي يوفر أفضل نتائج التنبؤ.

المصدر:

https://ernest-bonat.medium.com/rna-seq-gene-expression-classification-using-machine-learning-algorithms-de862e60bfd0

4) تعلم الآلة لعلم الجينوم Machine Learning For Genomics

(كيفية تحويل بيانات الجينوم لتناسب نماذج التعلم الآلي)

أصبح التعلم الآلي Machine learning شائعًا. ومع ذلك، فهي ليست حالة استخدام شائع في مجال المعلوماتية الحيوية Bioinformatics والبيولوجيا الحاسوبية Computational Biology. هناك عدد قليل جداً من الأدوات التي تستخدم تقنيات التعلم الآلي. تم تطوير معظم الأدوات على رأس الأساليب والخوارزميات القطعية.في هذه المقالة، سأقدم كيف يمكننا ترتيب بياناتنا بحيث يتم استخدامها بشكل فعال في نموذج التعلم الآلي.

مجالات التطبيق

هناك العديد من السيناريوهات في علم الجينوم genomics والتي قد نستخدم فيها التعلم الآلي. يمكن استخدام المجالات الرئيسية للتجميع Clustering والتصنيف Classification في علم الجينوم للقيام بمهام مختلفة. عدد قليل منهم على النحو التالي:

- التجميع (التعلم غير الخاضع للإشراف Unsupervised Learning)

1. تجميع الكونتيجات الميتاجينومية Binning of Metagenomics Contigs
2. تحديد البلازميدات والكروموسومات Identification of Plasmids and Chromosomes
3. التجميع يقرأ في الكروموسومات لتجميع أفضل Clustering reads into chromosomes for better assembly.
4. تجميع القراءات كمعالج أولي لتجميع القراءات Identification of Plasmids and Chromosomes

- التصنيف (التعلم الخاضع للإشراف Supervised Learning)

1. تصنيف التسلسلات الأقصر إلى فئات classes (الشعبة phylum، الجنس genus، الأنواع species، إلخ)
2. الاستدلال الوراثي للتسلسلات Phylogenetic inference of the sequences
3. الكشف عن البلازميدات والكروموسومات Detection of Plasmids and Chromosomes
4. البحث عن مناطق الترميز Finding coding regions
5. التنبؤ بالكروموسوم في الجينوميات البشرية Chromosome prediction in human genomics

يمكن أن تمتد القائمة إلى أبعد من ذلك بكثير، على الرغم من أنني قمت بإدراج بعض المجالات التي كانت لدي خبرة فيها.

تحويل البيانات واختيار النموذج

من بين الخطوتين، التحول transformation واختيار النموذج model selection، سأعتبر أن الأولى ذات أهمية أعلى. هذا لأنه بدون قاعدة صلبة لتمثيل البيانات، قد لا نحصل على أقصى استفادة من النموذج. ومع ذلك، فإن الحصول على بيانات نظيفة وغنية بالمعلومات يمكن أن يؤدي بشكل أفضل بشكل معقول حتى لو كان نموذجنا يبدو فقيرًا. يمكننا النظرفي البيانات ضمن فئتين.

البيانات المتسلسلة

تشير البيانات المتسلسلة Sequential Data إلى البيانات التي يتوافق فيها ترتيب البيانات التي يتم تغذيتها إلى النموذج مع الترتيب الفعلي للبياناتفي مجموعة البيانات. دعنا نلقي نظرة على المثال التالي.

1. **التنبؤ بمناطق ترميز البروتين Prediction of Protein Coding Regions**

   في هذا السيناريو، سننظرفي قواعد النوكليوتيدات المتتالية consecutive nucleotide bases وترتيبها. خلاف ذلك، لن يكون له معنى.في مثل هذا السيناريو، يمكننا تحويل البيانات إلى جمل تحتوي على كلمات متقطعة متتالية consecutive trimer words.

2. **التنبؤ بالتسمية التصنيفية Taxonomic Label Prediction**

   في هذا السيناريو، نحتاج إلى دقة أعلى حيث يمكن أن تكون ترددات قليل النوكليوتيد متشابهة جدًا بين الأنواع ولها تميز منخفض جدًا بين أنواع معينة. لذلك، سننظرفي إنشاء جمل k-mer لقيم k أعلى من 5 أو 7 كجمل.

دعونا نتأمل المثال التالي. سأستخدم النوعين Pseudomonas aeruginosa (CP007224.1) و Lactobacillus fermentum (AP008937.1) لإثبات التحول إلى جمل كلمة k-mer (أول 50 قاعدة معروضة).

```
Pseudomonas aeruginosa
TTTAAAGAGACCGGCGATTCTAGTGAAATCGAACGGGCAGGTCAATTTCCLactobacillus
fermentum
TTGACTGAGCTCGATTCTCTTTGGGAAGCGATCCAAAATTCATTCCGTAA
```

الآن، يمكننا تحويل هذا إلى كلمات مقتطعة trimer words لاظهار الناتج التالي.

```
Pseudomonas aeruginosa
TTT TTA TAA AAA AAG AGA GAG AGA GAC ACC CCG CGG GGC GCG CGA GAT ATT
TTC TCT CTA TAG AGT GTG TGA GAA AAA AAT ATC TCG CGA GAA AAC ACG CGG
GGG GGC GCA CAG AGG GGT GTC TCA CAA AAT ATT TTT TTC TCCLactobacillus
fermentum
TTG TGA GAC ACT CTG TGA GAG AGC GCT CTC TCG CGA GAT ATT TTC TCT CTC
```

```
TCT CTT TTT TTG TGG GGG GGA GAA AAG AGC GCG CGA GAT ATC TCC CCA CAA
AAA AAA AAT ATT TTC TCA CAT ATT TTC TCC CCG CGT GTA TAA
```

الآن بعد أن أصبح لدينا جمل من الكلمات، أصبحت المعالجة مشابهة لتلك الخاصة بتحليل المشاعر sentiment analysis. الفكرة الأساسية هي الحفاظ على طول الجملة، والذي يمكنك تحديده بناءً على متوسط طول التسلسلات.

نموذج مناسب للبيانات المتسلسلة

نظرًا لأن ترتيب k-mers مهمفي السيناريو أعلاه، يمكننا بسهولة استخدام الشبكة العصبية المتكررة Recurrent Neural Network (RNN) أو نموذج الذاكرة طويلة قصيرة المدى Long Short Term Memory model (LSTM). هذا بسبب قدرتهم على الحفاظ على ترتيب العناصر كعلاقة زمنية.

ومع ذلك،في السيناريو أعلاه، يجب استخدام الترميز tokenization متبوعًا بطبقة تضمين الكلمة word embedding layer. تأكد من اختيار المشفرات encoders والمميزات tokenizersفي ملف (تسلسل serialise) بحيث يكون لديك المشفر للتنبؤات لاحقًا. هنا kmer_strings هي الجمل التي أنشأتها والفئات classes هي التسمية المحددة.

```
tokenizer = Tokenizer()
tokenizer.fit_on_texts(kmer_strings)
vocab_size = len(tokenizer.word_index) + 1
encoded_docs = tokenizer.texts_to_sequences(kmer_strings)binarizer =
preprocessing.LabelBinarizer()
labels_encoded = binarizer.fit_transform(classes)
```

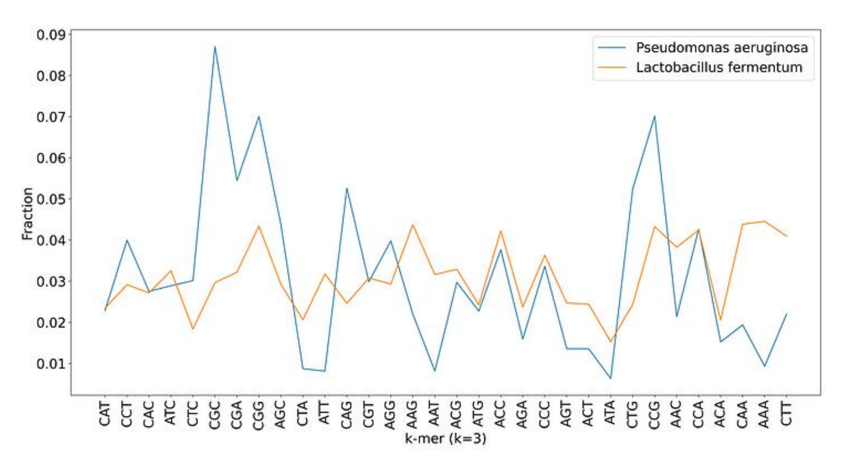
فيما يلي أحد النماذج التي استخدمتها لتصنيف التسلسل قبل بضعة أيام باستخدام Keras Sequential API. أقوم بتضمين الكلمات في 5 أبعاد من الأبعاد 32 الأولية (هناك 32 أداة تشذيب فريدة unique trimers تدمج المكملات العكسية للخيط السفلي lower strand).

```
model = Sequential()
model.add(Embedding(vocab_size, 5))
model.add(Bidirectional(LSTM(5)))
model.add(Dense(30, activation = 'relu'))
model.add(Dropout(0.2))
model.add(Dense(20, activation = 'relu'))
model.add(Dropout(0.2))
model.add(Dense(2, activation = 'softmax'))
```

إذا كان تصنيفك متعدد التسميات multi-label، فستحتاج إلى استخدام دالة التنشيط السيني sigmoid بدلاً من دالة softmax.

البيانات غير المرتبة

بالنسبة للسيناريوهات التي يكون فيها الارتباط بين التسلسلات المتشابهة أكثر أهمية من قابلية التعرف الفريدة للتسلسلات، فإننا نميل إلى استخدام البيانات غير المرتبة unordered data. وعادة ما يتم تمثيل هذهفي شكل متجهات تردد قليل النوكليوتيد oligonucleotide frequency vectors. على سبيل المثال، تبدو الترددات المعيارية normalized frequencies لـ Pseudomonas aeruginosa (CP007224.1) و Lactobacillus fermentum (AP008937.1) كما يلي.



يمكننا أن نرى أنهم يتبعون أنماطًا مختلفة تمامًا. لاحظ أننا فقدنا الترتيب الفعلي لأدوات التشذيب trimers ولكننا تركنا مع تمثيل لتحديد النوعين بشكل فريد. ومع ذلك، عند k = 3 من المحتمل جداً أن تصادمات بين الاصناف وثيقة الصلة.

الآن بعد أن رأيت البيانات، يمكنك استخدام نموذج مشابه لما يلي للقيام بالتصنيف.

```
model = Sequential()
model.add(Dense(30, activation = 'relu', input_shape=(32,)))
model.add(Dropout(0.2))
model.add(Dense(20, activation = 'relu'))
model.add(Dropout(0.2))
model.add(Dense(2, activation = 'softmax'))
model.compile(loss='binary_crossentropy', optimizer='adam')
```

لاحظ أن بُعد الإدخال الخاص بك سيزداد بقدرات 4 عند اتخاذ قرار بشأن k-mers أطول لهذا الغرض. علاوة على ذلك، سيستغرق حساب مثل هذه المتجهات الطويلة وقتًا طويلاً جداً.

عدد قليل من الأدوات الموجودة

الآن بعد أن رأيت كيف يمكن للمرء استخدام التسلسلات الجينية ذات الأطوال المتغيرةفي نموذج التعلم الآلي، اسمحوا لي أن أعرض بعض الأدوات التي تفعل ذلك بالفعل.

- PlasClass (تم النشر عام 2020 ، PLOS).

يستخدم الانحدار اللوجستي logistic regression على متجهات تردد k-mer لاكتشاف ما إذا كانت تنشأ من تسلسل بلازميد plasmid sequence أو مقطع كروموسومي chromosomal segment. هذه أداة قائمة على التصنيف الثنائي binary classification.

- PlasFlow (تم نشره عام 2018 ، Nucleic Acid Research).

تتنبأ هذه الأداة بتصنيف مستوى الشعبة phylum level classification وتتوقع ما إذا كان كونتيج contig معيناً هو بلازميد أو كروموسوم. يستخدم شبكة عصبية فوق متجهات تردد k-mer.

- MetaBCC-LR (تم النشر ، 2020 ، Bioinformatics).

يستخدم تضمين (t-SNE) Stochastic Neighbor Embedding الموزع على شكل t لتقليل القراءات الجينومية الطويلة لإجراء تجميع متجهات قاطعة للقراءات الميتاجينومية metagenomic reads' trimer vectors.

هناك العديد من الأدوات في مجالات مختلفة من البحث بخلاف تلك القليلة (التي استخدمتها والثالثة التي قمت بتأليفها). تُستخدم LSTMs في التنبؤ الجيني وكشف منطقة الترميز region detection.

المصدر:

https://towardsdatascience.com/machine-learning-for-genomics-c02270a51795

5) استكشاف عالم المعلوماتية الحيوية باستخدام التعلم الآلي Explore the world of Bioinformatics with Machine Learning

تحتوي المقالة على مقدمة موجزة للمعلوماتية الحيوية Bioinformatics وكيف يمكن استخدام خوارزمية تصنيف التعلم الآلي machine learning classification لتصنيف نوع السرطان في كل مريض من خلال تعبيراته الجينية gene expressions.

المعلوماتية الحيوية Bioinformatics هي مجال دراسة يستخدم الحساب لاستخراج المعرفة من البيانات البيولوجية biological data. وهو يشمل جمع البيانات وتخزينها واسترجاعها ومعالجتها ونمذجة البيانات لتحليلها أو تصورها أو التنبؤ بها من خلال تطوير الخوارزميات والبرامج.

يمكننا اقتباسها بطريقة أبسط "تتعامل المعلوماتية الحيوية مع المناهج الحسابية والرياضية لفهم البيانات البيولوجية biological data ومعالجتها".

إنه مجال متعدد التخصصات يتم فيه تطوير طرق حسابية جديدة لتحليل البيانات البيولوجية وإجراء اكتشافات بيولوجية. على سبيل المثال، هناك مهمتان نموذجيتان في علم الوراثة genetics وعلم الجينوم genomics هما عمليات التسلسل sequencing والتعليق التوضيحي annotating لمجموعة كاملة من الحمض النووي DNA للكائن الحي. في علوم الأعصاب neurosciences، تُستخدم تقنيات التصوير العصبي techniques neuroimaging، مثل التصوير المقطعي المحوسب computerized tomography (CT)، والتصوير المقطعي بالإصدار البوزيتروني positron emission tomography (PET)، والتصوير بالرنين المغناطيسي الوظيفي functional magnetic resonance imaging (fMRI)، والتصوير الموتر للانتشار diffusion tensor imaging (DTI)، لدراسة الأدمغة في الجسم الحي وفهم الأعمال الداخلية من الجهاز العصبي.

يفتح تطبيق التعلم الآلي Machine Learning على البيانات البيولوجية والتصوير العصبي neuroimaging data آفاقًا جديدة للهندسة الطبية الحيوية biomedical engineering: تحسين فهمنا للأمراض المعقدة مثل السرطان أو الاضطرابات العصبية التنكسية والنفسية. يمكن أن يؤدي التقدم في هذا المجال في النهاية إلى تطوير أدوات التشخيص الآلي والطب الدقيق، والذي يتكون من استهداف العلاجات الطبية المخصصة مع مراعاة التباين الفردي ونمط الحياة والبيئة.

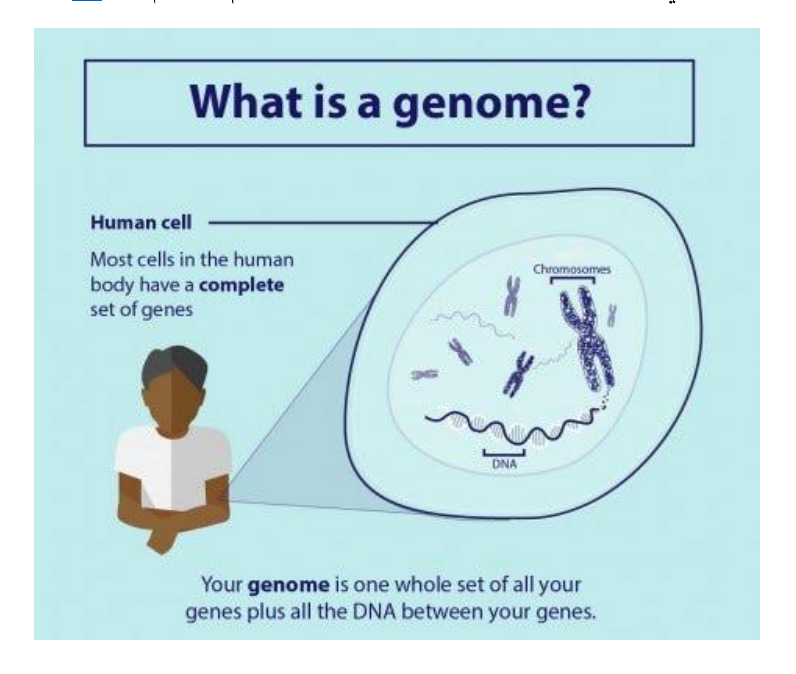
قبل ظهور خوارزميات التعلم الآلي، كان لابد من برمجة خوارزميات المعلوماتية الحيوية يدويًا بشكل صريح والتي ثبت أنها صعبة للغاية بالنسبة لمشاكل مثل التنبؤ ببنية البروتين protein structure prediction.

تمكّن تقنيات التعلم الآلي مثل التعلم العميق deep learning الخوارزمية من الاستفادة من التعلم التلقائي للميزات automatic feature learning مما يعني أنه بناءً على مجموعة البيانات وحدها، يمكن للخوارزمية أن تتعلم كيفية دمج ميزات متعددة لبيانات الإدخال في مجموعة أكثر تجريدًا من الميزات التي يمكن من خلالها إجراء مزيد من التعلم. يسمح هذا النهج متعدد الطبقات لأنماط التعلم في بيانات الإدخال لمثل هذه الأنظمة بعمل تنبؤات معقدة للغاية عند تدريبها على مجموعات البيانات الكبيرة. في السنوات الأخيرة، ارتفع حجم وعدد مجموعات البيانات البيولوجية المتاحة بشكل كبير، مما مكّن باحثي المعلوماتية الحيوية من الاستفادة من خوارزميات التعلم الآلي هذه.

تم تطبيق التعلم الآلي على ست مجالات بيولوجية: علم الجينوم Genomics، وعلم البروتينات Proteomics، والمصفوفات الدقيقة Microarrays، وبيولوجيا الأنظمة Systems biology، وتشخيص السكتات الدماغية Stroke diagnosis، والتنقيب في النص Text mining.

علم الجينوم

إنه مجال متعدد التخصصات في علم الأحياء يركز على هيكل structure ووظيفة function وتطور evolution وتعيين mapping وتحرير editing الجينوم genomes. الجينوم هو مجموعة كاملة من الحمض النووي DNA للكائن الحي، بما في ذلك جميع جيناته. هناك حاجة متزايدة لتطوير أنظمة التعلم الآلي التي يمكن أن تحدد تلقائيًا موقع الجينات المشفرة للبروتين protein-encoding genes ضمن تسلسل DNA معين وتعرف هذه المشكلة في علم الأحياء الحسابي computational biology بالتنبؤ الجيني gene prediction. لمعرفة المزيد عن علم الجينوم انقر هنا.



علم البروتينات

علم البروتينات Proteomics هو دراسة واسعة النطاق للبروتينات proteomes. البروتين proteome هو مجموعة من البروتينات يتم إنتاجهافي كائن حي أو نظام أو سياق بيولوجي.

تكتسب البروتينات، وهي سلاسل من الأحماض الأمينية amino acids، الكثير من وظيفتها من طي البروتين protein folding حيث تتوافق مع بُنية ثلاثية الأبعاد. يتكون هذا الهيكل من عدد من طبقات الطي، بمافي ذلك البُنية الأساسية primary structure (أي السلسلة المسطحة للأحماض الأمينية flat string of amino acids)، والبنية الثانوية secondary structure (حلزونات ألفا alpha helices وصفائح بيتا beta sheets)، والبُنية الثلاثية tertiary structure، والهيكل الربعي quarternary structure.

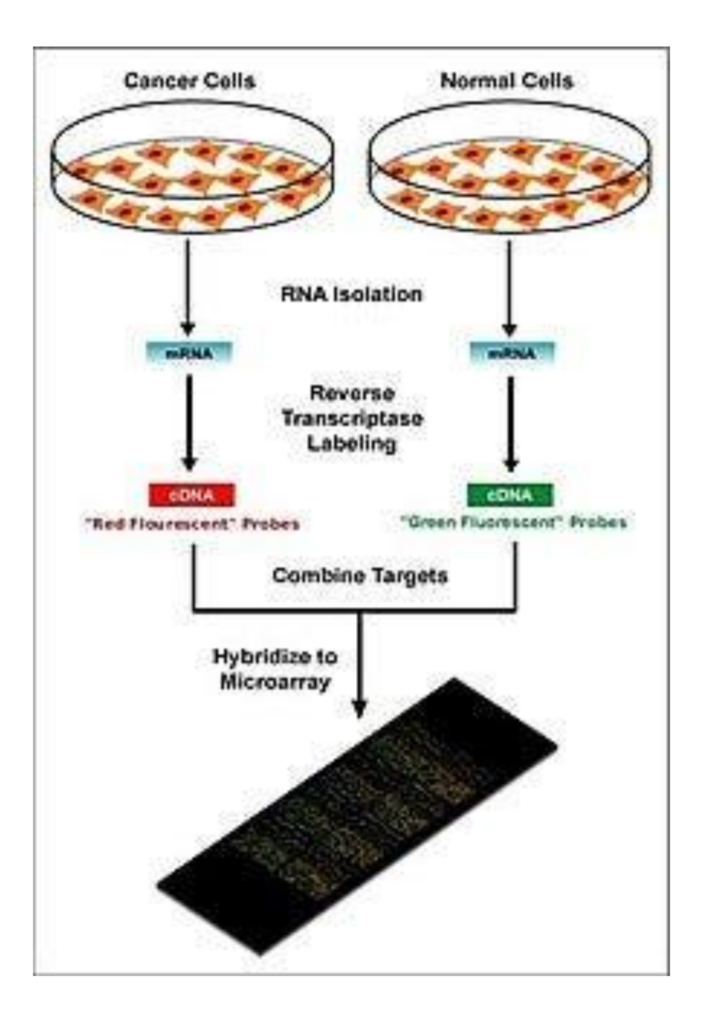
يعد التنبؤ بالبنية الثانوية للبروتين Protein secondary structure prediction هو المحور الرئيسي لهذا الحقل الفرعي حيث يتم تحديد طيات البروتين الإضافية protein foldings (البنى الثلاثية والرباعية tertiary and quaternary structures) بناءً على البُنية الثانوية. يعد حل البُنية الحقيقية للبروتين عملية مكلفة للغاية وتستغرق وقتًا طويلاً، مما يزيد من الحاجة إلى أنظمة يمكنها التنبؤ بدقة ببُنية البروتين عن طريق تحليل تسلسل الأحماض الأمينية مباشرةً. قبل التعلم الآلي، كان الباحثون بحاجة إلى إجراء هذا التنبؤ يدويًا.

تستخدم الحالة الحاليةفي التنبؤ بالهيكل الثانوي نظامًا يسمى DeepCNF (الحقول العصبية التلافيفية العميقة deep convolutional neural fields) الذي يعتمد على نموذج التعلم الآلي للشبكات العصبية الاصطناعية لتحقيق دقة تقارب 84٪ عند تكليفه بتصنيف الأحماض الأمينية من تسلسل البروتين إلى واحدة من ثلاث فئات هيكلية (حلزون helix، صفيحة sheet، أو ملف coil).

المصفوفات الدقيقة

تُستخدم المصفوفات الدقيقة Microarrays، وهي نوع من المعامل على رقاقة lab-on-a-chip، لتجميع البيانات تلقائيًا حول كميات كبيرة من المواد البيولوجية. يمكن أن يساعد التعلم الآليفي تحليل هذه البيانات، وقد تم تطبيقه لتحديد نمط التعبير expression pattern، والتصنيف classification، وتحريض الشبكة الجينية genetic network induction.

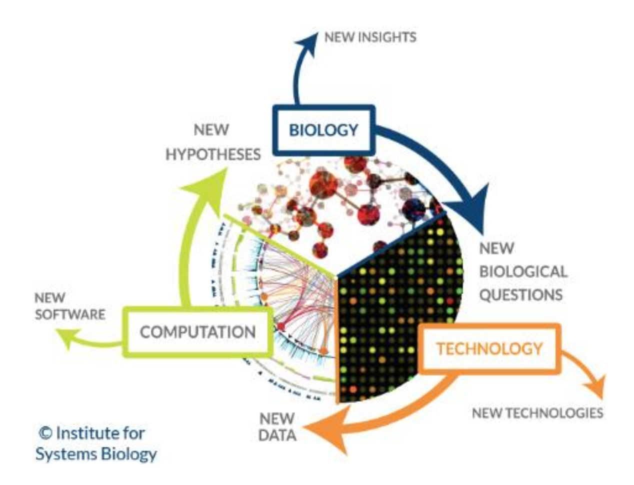
هذه التقنية مفيدة بشكل خاص لرصد تعبير الجينات داخل الجينوم، مما يساعدفي تشخيص أنواع مختلفة من السرطان بناءً على الجينات التي يتم التعبير عنها. تتمثل إحدى المشكلات الرئيسيةفي هذا المجالفي تحديد الجينات التي يتم التعبير عنها بناءً على البيانات التي تم جمعها.



يقدم التعلم الآلي حلاً محتملاً لهذه المشكلة حيث يمكن استخدام طرق تصنيف مختلفة لإجراء هذا التعريف. الطرق الأكثر شيوعًا هي شبكات وظائف الأساس الشعاعي radial basis function networks، والتعلم العميق deep learning، وتصنيف بايزي Bayesian classification، وأشجار القرار decision trees، والغابات العشوائية random forest.

بيولوجيا الأنظمة

يركز بيولوجيا الأنظمة Systems biology على دراسة السلوكيات الناشئة من التفاعلات المعقدة للمكونات البيولوجية البسيطةفي النظام. يمكن أن تشتمل هذه المكونات على جزيئات مثل DNA وRNA والبروتينات proteins والمستقلبات metabolites.



تم استخدام التعلم الآلي للمساعدةفي نمذجة هذه التفاعلات المعقدةفي الأنظمة البيولوجيةفي مجالات مثل الشبكات الجينية genetic networks وشبكات نقل الإشارات signal transduction networks والمسارات الأيضية metabolic pathways. تعد النماذج الرسومية الاحتمالية Probabilistic graphical models، وهي تقنية تعلم آلي لتحديد البنية بين المتغيرات المختلفة، واحدة من أكثر الطرق شيوعًا لنمذجة الشبكات الجينية. بالإضافة إلى ذلك، تم تطبيق التعلم الآلي على مشاكل بيولوجيا الأنظمة مثل تحديد مواقع ربط عامل النسخ transcription factor binding sites باستخدام تقنية تُعرف باسم تحسين سلسلة ماركوف Markov chain optimization. تم استخدام الخوارزميات الجينية Genetic algorithms، وتقنيات التعلم الآلي التي تستند إلى عملية التطور الطبيعية، لنمذجة الشبكات الجينية والهياكل التنظيمية.

تشخيص السكتة الدماغية

تُستخدم طرق التعلم الآلي لتحليل بيانات التصوير العصبي للمساعدةفي تشخيص السكتة الدماغية. غالبًا ما يتم استخدام أساليب الشبكة العصبية التلافيفية ثلاثية الأبعاد Three-dimensional Convolutional Neural Network (CNN) وآلة المتجهات الداعمة (Support Vector Machines (SVM).

التنقيب في النصوص

أدت الزيادةفي المنشورات البيولوجية المتاحة إلى قضية زيادة صعوبة البحث من خلال وتجميع جميع المعلومات المتاحة ذات الصلة حول موضوع معين عبر جميع المصادر. تُعرف هذه المهمة باسم استخراج المعرفة knowledge extraction. يعد هذا ضروريًا لجمع البيانات البيولوجية والتي يمكن بدورها إدخالهافي خوارزميات التعلم الآلي لتوليد معرفة بيولوجية جديدة. يمكن استخدام التعلم الآلي

لمهمة استخراج المعرفة هذه باستخدام تقنيات مثل معالجة اللغة الطبيعية Natural Language Processing (NLP) لاستخراج معلومات مفيدة من التقارير التي ينشئها الإنسانفي قاعدة بيانات.

تم تطبيق هذه التقنية على البحث عن أهداف دوائية جديدة، حيث تتطلب هذه المهمة فحص المعلومات المخزنةفي قواعد البيانات والمجلات البيولوجية. غالبًا لا تعكس التعليقات التوضيحية للبروتينات Annotations of proteinsفي قواعد بيانات البروتين المجموعة الكاملة المعروفة للمعرفة لكل بروتين، لذلك يجب استخراج معلومات إضافية من الأدبيات الطبية الحيوية. تم تطبيق التعلم الآلي على التعليق التوضيحي التلقائي automatic annotation لوظيفة الجينات والبروتينات، وتحديد التوطين الخلوي subcellular localization للبروتين، وتحليل صفائف تعبير الحمض النووي analysis of DNA-expression arrays، وتحليل تفاعل البروتين على نطاق واسع large-scale protein interaction analysis، وتحليل تفاعل الجزيء molecule interaction analysis.

يتيح لنا الآن تنفيذ خوارزمية (SVM)في مجموعة بيانات المعلوماتية الحيوية ومعرفة كيفية عملها.

التصنيف الجزيئي للسرطان عن طريق مراقبة التعبير الجيني باستخدام (SVM)

على الرغم من تحسن تصنيف السرطان خلال الثلاثين عامًا الماضية، لم يكن هناك نهج عام لتحديد فئات السرطان الجديدة (اكتشاف الفئة class discovery) أو لتخصيص الأورام tumors لفئات معروفة (التنبؤ بالفئة class prediction). تأتي مجموعة البيانات من دراسة إثبات المفهوم التي نُشرت في 1999 من قبل Golub et al. لقد أظهر كيف يمكن تصنيف حالات السرطان الجديدة عن طريق مراقبة التعبير الجيني gene expression monitoring (عبر مصفوفة الحمض النووي الدقيقة DNA microarray) وبالتالي قدم نهجًا عامًا لتحديد فئات السرطان الجديدة وتخصيص الأورام لفئات معروفة.

الهدف هو تصنيف المرضى الذين يعانون من سرطان الدم النخاعي الحاد acute myeloid leukemia (AML) وسرطان الدم الليمفاوي الحاد lymphoblastic leukemia (ALL) باستخدام خوارزمية SVM.

يمكن تنزيل مجموعة البيانات من Kaggle.

لنبدأ البرمجة:

ابدأ بتحميل 3 مكتبات أساسية للبايثون:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

قم بتحميل مجموعات البيانات:

```
Train_Data =
pd.read_csv("/.../bioinformatics/data_set_ALL_AML_train.csv")
Test_Data =
pd.read_csv("/.../bioinformatics/data_set_ALL_AML_independent.csv")
labels = pd.read_csv("/.../bioinformatics/actual.csv", index_col =
'patient')Train_Data.head()
```

	Gene Description	Gene Accession Number	1	call	2	call.1	3	call.2	4	call.3	...	29	call.33	30	call.34	31	call.35	32	call.36	33	call.37
0	AFFX-BioB-5_at (endogenous control)	AFFX-BioB-5_at	-214	A	-139	A	-76	A	-135	A	...	15	A	-318	A	-32	A	-124	A	-135	A
1	AFFX-BioB-M_at (endogenous control)	AFFX-BioB-M_at	-153	A	-73	A	-49	A	-114	A	...	-114	A	-192	A	-49	A	-79	A	-186	A
2	AFFX-BioB-3_at (endogenous control)	AFFX-BioB-3_at	-58	A	-1	A	-307	A	265	A	...	2	A	-95	A	49	A	-37	A	-70	A
3	AFFX-BioC-5_at (endogenous control)	AFFX-BioC-5_at	88	A	283	A	309	A	12	A	...	193	A	312	A	230	P	330	A	337	A
4	AFFX-BioC-3_at (endogenous control)	AFFX-BioC-3_at	-295	A	-264	A	-376	A	-419	A	...	-51	A	-139	A	-367	A	-188	A	-407	A

بيانات التدريب

```
Test_Data.head()
```

	Gene Description	Gene Accession Number	39	call	40	call.1	42	call.2	47	call.3	...	65	call.29	66	call.30	63	call.31	64	call.32	62	call.33
0	AFFX-BioB-5_at (endogenous control)	AFFX-BioB-5_at	-342	A	-87	A	22	A	-243	A	...	-62	A	-58	A	-161	A	-48	A	-176	A
1	AFFX-BioB-M_at (endogenous control)	AFFX-BioB-M_at	-200	A	-248	A	-153	A	-218	A	...	-198	A	-217	A	-215	A	-531	A	-284	A
2	AFFX-BioB-3_at (endogenous control)	AFFX-BioB-3_at	41	A	262	A	17	A	-163	A	...	-5	A	63	A	-46	A	-124	A	-81	A
3	AFFX-BioC-5_at (endogenous control)	AFFX-BioC-5_at	328	A	295	A	276	A	182	A	...	141	A	95	A	146	A	431	A	9	A
4	AFFX-BioC-3_at (endogenous control)	AFFX-BioC-3_at	-224	A	-226	A	-211	A	-289	A	...	-256	A	-191	A	-172	A	-496	A	-294	A

بيانات الاختبار

حول مجموعة البيانات:

- يمثل كل صف جينًا مختلفًا.
- العمودان 1 و 2 عبارة عن أوصاف حول هذا الجين.
- كل عمود مرقم هو مريض في بيانات التسمية label data.
- لكل مريض 7129 قيمة تعبير جيني gene expression values ـ أي لكل مريض قيمة واحدة لكل جين.
- تحتوي بيانات التدريب على قيم التعبير الجيني للمرضى من 1 إلى 38.
- تحتوي بيانات الاختبار على قيم التعبير الجيني للمرضى من 39 إلى 72.

تحقق الآن من القيم الخاليةفي مجموعتي البيانات (ليس لدينا أي قيم خاليةفي مجموعات البيانات هذه).

```
print(Train_Data.isna().sum().max())
print(Test_Data.isna().sum().max())
```

الآن قم بإسقاط العمود "call" من بيانات التدريب والاختبار لأنه لا يحتوي على أي صلة إحصائية.

```
cols = [col for col in Test_Data.columns if 'call' in col]
test = Test_Data.drop(cols, 1)
cols = [col for col in Train_Data.columns if 'call' in col]
train = Train_Data.drop(cols, 1)
```

انضم الآن إلى جميع مجموعات البيانات وقم بتبديل البيانات النهائية المرتبطة.

```
patients = [str(i) for i in range(1, 73, 1)]
df_all = pd.concat([train, test], axis = 1)[patients]
df_all = df_All.T
```

	0	1	2	3	4	5	6	7	8	9	...	7119	7120	7121	7122	7123	7124	7125	7126	7127	7128
1	-214	-153	-58	88	-295	-558	199	-176	252	206	...	185	511	-125	389	-37	793	329	36	191	-37
2	-139	-73	-1	283	-264	-400	-330	-168	101	74	...	169	837	-36	442	-17	782	295	11	76	-14
3	-76	-49	-307	309	-376	-650	33	-367	206	-215	...	315	1199	33	168	52	1138	777	41	228	-41
4	-135	-114	265	12	-419	-585	158	-253	49	31	...	240	835	218	174	-110	627	170	-50	126	-91
5	-106	-125	-76	168	-230	-284	4	-122	70	252	...	156	649	57	504	-26	250	314	14	56	-25

5 rows × 7129 columns

بعد التحويل، تم تحويل الصفوف إلى أعمدة (7129 عمود / معلم)

الآن قم بتحويل عمود المريض "patient" إلى قيمة رقمية وإنشاء متغيرات وهمية dummy variables (تحويل الفئات categories إلى قيم رقمية numeric values) نظرًا لأن "السرطان cancer " هو عمود فئوي به فئتان (ALL، AML).

```
df_all["patient"] = pd.to_numeric(patients)
labels["cancer"]= pd.get_dummies(Actual.cancer, drop_first=True)
```

اربط الآن إطارات البيانات df_all والتسمياتفي عمود المريض.

```
Data = pd.merge(df_all, labels, on="patient")
Data.head()
```

	0	1	2	3	4	5	6	7	8	9	...	7121	7122	7123	7124	7125	7126	7127	7128	patient	cancer
0	-214	-153	-58	88	-295	-558	199	-176	252	206	...	-125	389	-37	793	329	36	191	-37	1	0
1	-139	-73	-1	283	-264	-400	-330	-168	101	74	...	-36	442	-17	782	295	11	76	-14	2	0
2	-76	-49	-307	309	-376	-650	33	-367	206	-215	...	33	168	52	1138	777	41	228	-41	3	0
3	-135	-114	265	12	-419	-585	158	-253	49	31	...	218	174	-110	627	170	-50	126	-91	4	0
4	-106	-125	-76	168	-230	-284	4	-122	70	252	...	57	504	-26	250	314	14	56	-25	5	0

خطوتنا التالية هي إنشاء متغيرين X (مصفوفة المتغيرات المستقلة independent variables) و y (متجه المتغير التابع dependent variable).

```
X, y = Data.drop(columns=["cancer"]), Data["cancer"]
```

بعد ذلك، قمنا بتقسيم 75٪ من البيانات إلى مجموعة تدريب بينما يتم اختبار 25٪ من البيانات. المتغير test_size هو المكان الذي نحدد فيه بالفعل نسبة مجموعة الاختبار.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test =  train_test_split(X,y,test_size
= 0.25, random_state= 0)
```

الخطوة التالية هي تسوية البيانات normalize لأننا إذا نظرنا عن كثب إلى البيانات، فإن نطاق قيم المتغيرات المستقلة يختلف كثيرًا. لذلك عندما تختلف القيم كثيرًافي المتغيرات المستقلة، فإننا نستخدم تحجيم الميزة feature scaling بحيث تظل جميع القيم في النطاق القابل للمقارنة.

```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
```

عدد الأعمدة / الميزات التي كنا نعمل معها ضخم. لدينا 72 صفًا و 7129 عمودًا. نحتاج أساسًا إلى تقليل عدد الميزات (تقليل الابعاد Dimentioanlity Reduction) لإزالة إمكانية لعنة الأبعاد Curse of Dimensionality.

لتقليل عدد الأبعاد / الميزات، سنستخدم خوارزمية تقليل الأبعاد الأكثر شيوعًا، مثل PCA (تحليل المكون الرئيسي Principal Component Analysis).

لإجراء PCA، يتعين علينا اختيار عدد الميزات / الأبعاد التي نريدهافي بياناتنا.

```
from sklearn.decomposition import PCA
pca = PCA()
X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)
total=sum(pca.explained_variance_)
k=0
current_variance=0
while current_variance/total < 0.90:
    current_variance += pca.explained_variance_[k]
    k=k+1
```

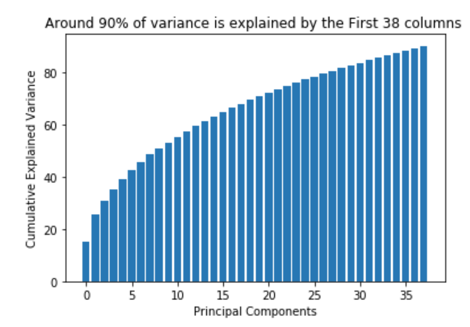
يعطي الكود أعلاه k = 38.

الآن دعونا نأخذ k = 38 ونطبق PCA على متغيراتنا المستقلة.

```
from sklearn.decomposition import PCA
pca = PCA(n_components = 38)
X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test) cum_sum =
pca.explained_variance_ratio_.cumsum()
cum_sum = cum_sum*100
plt.bar(range(38), cum_sum)
plt.ylabel("Cumulative Explained Variance")
plt.xlabel("Principal Components")
```

```
plt.title("Around 90% of variance is explained by the First 38 columns ")
```

```
Text(0.5, 1.0, 'Around 90% of variance is explained by the First 38 columns ')
```



ملاحظة: يمكن أن يؤدي PCA إلى انخفاض في أداء النموذج في مجموعات البيانات مع عدم وجود ارتباط correlation بين الميزات أو انخفاضها أو عدم تلبية افتراضات الخطية linearity.

تتمثل الخطوة التالية في ملاءمة بياناتنا في خوارزمية Support Vector Machine (SVM) ولكن قبل القيام بذلك سنقوم بتحسين المعلمات الفائقة Hyperparameter.

تحسين أو ضبط المعلمات الفائقة Hyperparameter optimization or tuning هي مشكلة اختيار مجموعة من المعلمات الفائقة المثلى لخوارزمية التعلم. المعلمة الفائقة هي معلمة تُستخدم قيمتها للتحكم في عملية التعلم. على النقيض من ذلك، يتم تعلم قيم المعلمات الأخرى.

سنستخدم GridSearchCV من sklearn لاختيار أفضل المعلمات الفائقة.

```
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVCparameters = [{'C': [1, 10, 100, 1000],
'kernel': ['linear']},
              {'C': [1, 10, 100, 1000], 'kernel': ['rbf'], 'gamma':
[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]}]search =
GridSearchCV(SVC(), parameters, n_jobs=-1, verbose=1)
search.fit(X_train, y_train)
```

تحقق الآن من أفضل المعلمات لخوارزمية SVM الخاصة بنا.

```
best_parameters = search.best_estimator_
```

```
SVC(C=1, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
    kernel='linear', max_iter=-1, probability=False, random_state=None,
    shrinking=True, tol=0.001, verbose=False)
```

الآن دعنا ندرب نموذج تصنيف SVM الخاص بنا.

```
model = SVC(C=1, cache_size=200, class_weight=None, coef0=0.0,
```

```
    decision_function_shape='ovr', degree=3,
gamma='auto_deprecated',
    kernel='linear', max_iter=-1, probability=False,
random_state=None,
    shrinking=True, tol=0.001, verbose=False)

model.fit(X_train, y_train)
```

حان الوقت لبعض التوقعات:

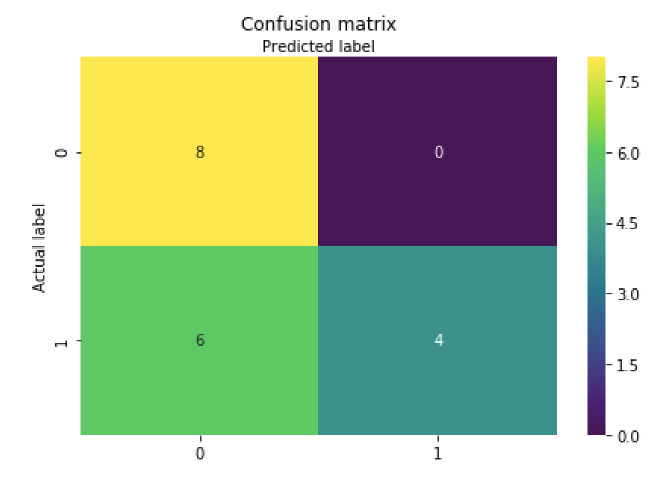
```
y_pred=model.predict(X_test)
```

تقييم أداء النموذج:

```
from sklearn.metrics import accuracy_score, confusion_matrixfrom
sklearn import metrics
print('Accuracy Score:',round(accuracy_score(y_test, y_pred),2))
#confusion matrix
cm = confusion_matrix(y_test, y_pred)
Output:
Accuracy Score: 0.67
```

مصفوفة الارتباك Confusion matrix وتصورها باستخدام خريطة الحرارة Heatmap.

```
class_names=[1,2,3]
fig, ax = plt.subplots()from sklearn.metrics import
confusion_matrix
import seaborn as snscm = confusion_matrix(y_test,
y_pred)class_names=['ALL', 'AML']
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
sns.heatmap(pd.DataFrame(cm), annot=True, cmap="viridis" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```



حسنًا، يوضح هذا المثال أنه إذا توقعت للتو أن كل مريض مصاب بمرض (AML)، فستكون على صواب أكثر من الخطأ.

لذلك توقع نموذج تصنيف SVM مرضى السرطان بدقة 67٪ وهو بالطبع ليس جيدًا. ما يمكنك القيام به هو تجربة مصنفات مختلفة مثل Random Forest و K-NN و Gradient Boosting و xgboost وما إلى ذلك ومقارنة الدقة لكل نموذج.

الاستنتاج

لذلكفي هذه المقالة، رأينا كيف يمكن استخدام خوارزمية تصنيف التعلم الآلي للتنبؤ بالسرطان لدى المريض.

في النهاية، أعتقد أن التعلم الآلي سوف يزدهر حقًا، فسوف يتحول إلى بيانات معلوماتية حيوية أفضل. تتمتع بيانات الصحة والمعلوماتية الحيويةفي الوقت الحالي بقدرة إحصائية ضعيفة جدًا. إما أنها عادة ما تكون ذات إشارة ضعيفة (علم الجينوم)، أو ضوضاء noise/تحيز bias مرتفعة(السجلات الصحية الإلكترونية)، أو أحجام عينات صغيرة.

المصدر

https://www.kdnuggets.com/2019/09/explore-world-bioinformatics-machine-learning.html

6) تحليل جينوم فيروس كورونا COVID-19 باستخدام Biopython

Coronavirus COVID-19 Genome Analysis using Biopython

لذلك في هذه المقالة، سوف نفسر ونحلل بيانات تسلسل الحمض النووي DNA لـ COVID-19 ونحاول الحصول على العديد من الأفكار المتعلقة بالبروتينات التي تتكون منها. سنقارن لاحقًا الحمض النووي لـ COVID-19 بمتلازمة الشرق الأوسط التنفسية MERS والسارس SARS وسنفهم العلاقة بينهما.

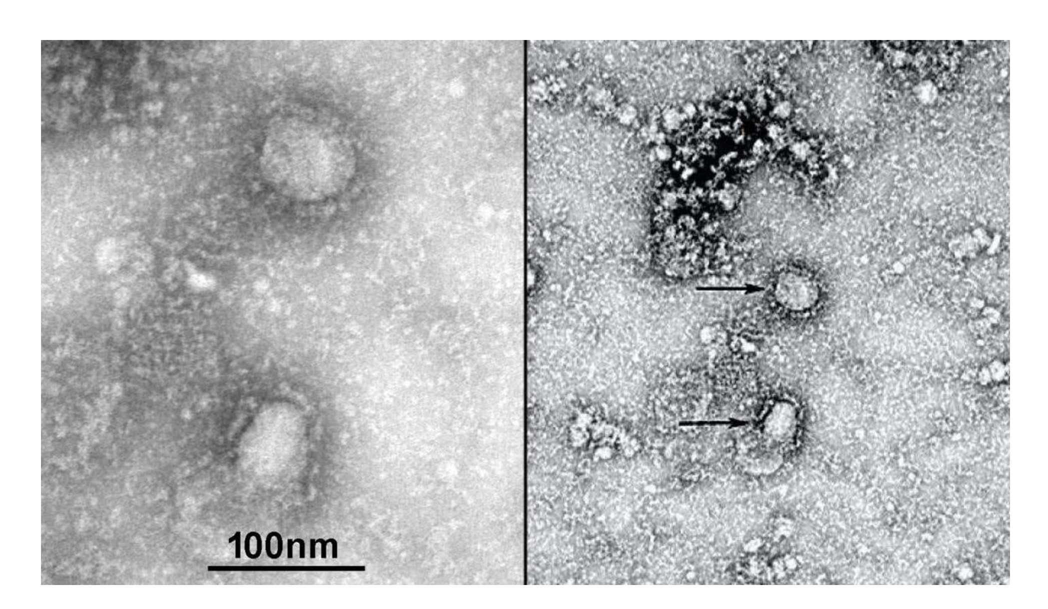
يمثل مرض فيروس كورونا المستجد العالمي COVID-19 الناشئ عن فيروس كورونا الجديد المتلازمة التنفسية الحادة الوخيمة 2 (SARS-CoV-2) انفجارات حرجة للصحة العامة العالمية والاقتصاد منذ أن تم تحديده في أواخر ديسمبر 2019 في الصين.

فيروسات كورونا Coronaviruses هي عائلة كبيرة من الفيروسات التي يمكن أن تسبب أمراضًا تتراوح شدتها على نطاق واسع. ظهر أول مرض خطير معروف ناجم عن فيروس كورونا مع وباء المتلازمة التنفسية الحادة الوخيمة (سارس) Severe Acute Respiratory Syndrome (SARS) عام 2003 في الصين. نشأت الفاشية الثانية للمرض الحاد في عام 2012 في المملكة العربية السعودية مع متلازمة الشرق الأوسط التنفسية (ميرس) Middle East Respiratory Syndrome (MERS). والآن التفشي المستمر لـ COVID-19.

"من خلال مقارنة بيانات تسلسل الجينوم المتاحة لسلالات فيروس كورونا المعروفة، يمكننا أن نقرر بحزم أن COVID-19 نشأ من خلال العمليات الطبيعية"، كما قال كريستيان أندرسن، دكتوراه، أستاذ مشارك في علم المناعة وعلم الأحياء المجهرية في أبحاث Scripps والمؤلف المقابل في الورقة.

لذلك في هذه المقالة، سوف نفسر ونحلل بيانات تسلسل الحمض النووي DNA لـ COVID-19 ونحاول الحصول على العديد من الأفكار المتعلقة بالبروتينات التي تتكون منها. سنقارن لاحقًا الحمض النووي لـ COVID-19 بمتلازمة الشرق الأوسط التنفسية MERS والسارس SARS وسنفهم العلاقة بينهما.

إذا كنت جديدًا في علم الجينوم، فإنني أوصيك بضرورة مراجعة المقالة إزالة الغموض عن تسلسل الحمض النووي باستخدام التعلم الآلي و Python قبل المضي قدمًا للحصول على فهم أساسي لتحليل بيانات الحمض النووي.



صور بالمجهر الإلكتروني لسلالة فيروس كورونا القاتلة

فيروسات كورونا هي أعضاءفي عائلة من الفيروسات المغلفة التي تتكاثرفي سيتوبلازم الخلايا المضيفة الحيوانية. يتم تحديدها من خلال وجود جينوم الحمض النووي الريبي أحادي الجديلة -single stranded plus-sense RNA genome (ssRNA(+)) تصنيف للفيروسات) يبلغ طوله حوالي 30 كيلو بايت وله هيكل 5 غطاء و 3 مسار متعدد الأدينيل. (أكبر فيروس معروف).

الآن دعونا نتعامل مع بيانات تسلسل الحمض النووي COVID2—19 باستخدام Python.

لتبدأ، سيساعدك تثبيت حزم Python مثل Biopython وsquiggle عند التعامل مع بيانات التسلسل البيولوجي biological sequence dataفي Python.

```
pip install biopython
pip install Squiggle
```

قم بتحميل المكتبات الأساسية:

```
import numpy as np
import pandas as pd
pd.plotting.register_matplotlib_converters()
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import os
```

يمكن تنزيل مجموعة البيانات من Kaggle.

سنستخدم Bio.SeqIO من Biopython لتحليل بيانات تسلسل الحمض النووي (fasta). يوفر واجهة موحدة بسيطة لإدخال وإخراج تنسيقات ملفات تسلسلية متنوعة.

```
from Bio import SeqIOfor sequence in
SeqIO.parse('/coronavirus/MN908947.fna', "fasta"):
print(sequence.seq)
print(len(sequence),'nucliotides')
```

لذلك ينتج التسلسل sequence وطول المتسلسلة length of the sequence.

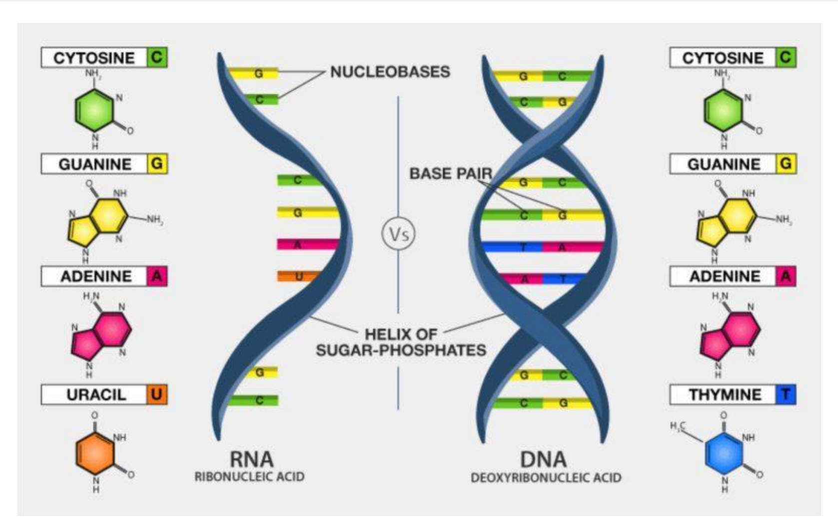
```
GCAATGGATACAACTAGCTACAGAGAAGCTGCTTGTTGTCATCTCGCAAAGGCTCTCAATGACTTCA
GTAACTCAGGTTCTGATGTTCTTTACCAACCACCACAAACCTCTATCACCTCAGCTGTTTTGCAGAG
TGGTTTTAGAAAAATGGCATTCCCATC......AGAATGACAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAA29903 nucliotides
```

تحميل تسلسل الحمض النووي التكميلي Complementary DNA Sequenceفي ملف قابل للتراصف alignable file.

```
from Bio.SeqRecord import SeqRecord
from Bio import SeqIO
DNAsequence = SeqIO.read('/coronavirus/MN908947.fna', "fasta")
```

`SeqIO.read()` سوف ينتج المعلومات الأساسية المتعلقة بالتسلسل.

```
SeqRecord(seq=Seq('ATTAAAGGTTTATACCTTCCCAGGTAACAAACCAACCAACTTTCGATC
TCTTGT...AAA', SingleLetterAlphabet()), id='MN908947.3',
name='MN908947.3', description='MN908947.3 Severe acute respiratory
syndrome coronavirus 2 isolate Wuhan-Hu-1, complete genome',
dbxrefs=[])
```



نظرًا لأن تسلسل الإدخال هو FASTA (DNA) ، و Coronavirus هو نوع من الفيروسات من نوع RNA ، فنحن بحاجة إلى:

- نسخ (Transcribe) DNA إلى RNA (ATTAAAGGTT... => AUUAAAGGUU...)
- ترجمة (Translate) RNA إلى تسلسل الأحماض الأمينية (...AUUAAAGGUU => IKGLYLPR * Q ...)

في السيناريو الحالي، يبدأ ملف fna. بـ ATTAAAGGTT ، ثم نسمي ()transcribe لذلك يتم استبدال T (الثايمين thymine) بـ U (يوراسيل uracil) ، لذلك نحصل على تسلسل RNA الذي يبدأ بـ AUUAAAGGUU.

```
DNA = DNAsequence.seq#Convert DNA into mRNA Sequence
mRNA = DNA.transcribe() #Transcribe a DNA sequence into RNA.
print(mRNA)
print('Size : ',len(mRNA))
```

سيحول ()transcribe الحمض النووي DNA إلى RNA.

```
UAUUUUAGUGGAGCAAUGGAUACAACUAGCUACAGAGAAGCUGCUUGUUGUCAUCUCGCAAAGGCUC
UCAAUGACUUCAGUAACUCAGGUUC...UAAUAGCUUCUUAGGAGAAUGACAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAA
Size : 29903
```

الفرق بين DNA و mRNA هو أن القواعد T (للثايمين) يتم استبدالها بـ U (لـليوراسيل).

بعد ذلك، نحتاج إلى ترجمة تسلسل mRNA إلى تسلسل الأحماض الأمينية amino-acid sequence باستخدام طريقة ()translate ، نحصل على شيء مثل IKGLYLPR * Q (يسمى كودون الايقاف (STOP codon)، وهو فعال كفاصل للبروتينات).

```
Amino_Acid = mRNA.translate(table=1, cds=False)
print('Amino Acid', Amino_Acid)
print("Length of Protein:",len(Amino_Acid))
print("Length of Original mRNA:",len(mRNA))
```

في الناتج أدناه، يتم فصل الأحماض الأمينية بعلامة *.

```
Amino Acid :
IKGLYLPR*QTNQLSISCRSVL*TNFKICVAVTRLHA*CTHAV*LITNYCR*QDTSNSSIFCRLLTV
SSVLQPIISTSRFRPGVTER*DGEPCPWFQRENTRPTQFACFTGSRRARTWLWRLRGGGLIRGTSTS
*RWHLWLSRS*KRRFAST*TALCVHQTFGCSNCTSWSCYG...*SHIAIFNQCVTLGRT*KSHHIFT
EATRSTIECTVNNARESCLYGRALMCKINFSSAIPM*F**LLRRMTKKKKKKKKKLength of
Protein : 9967
Length of Original mRNA : 29903
```

في السيناريو الخاص بنا، يبدو التسلسل كما يلي: IKGLYLPR * QTNQLSISCRSVL * TNFKICVAVTRLHA، حيث:

يقوم IKGLYLPR بترميز البروتين الأول (كل حرف يشفر حمض أميني واحد) يشفر QTNQLSISCRSVL البروتين الثاني، وهكذا.

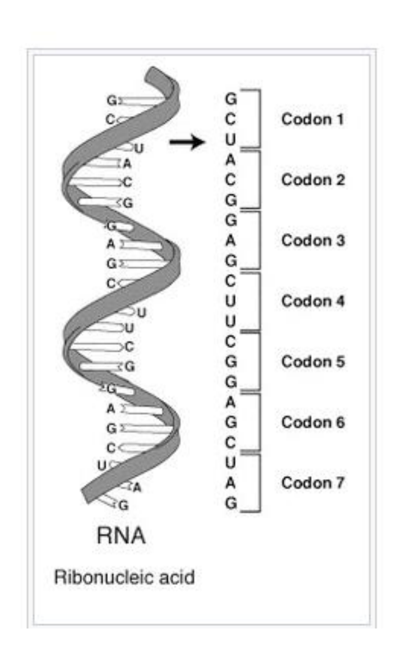
لاحظ أن هناك تسلسلات أقل في البروتين من mRNA وذلك لأن 3 mRNA تُستخدم لإنتاج وحدة فرعية واحدة من البروتين، تُعرف باسم الأحماض الأمينية amino acid، باستخدام جدول الكودون codon table الموضح أدناه. * تُستخدم للإشارة إلى كودون الإيقاف،في هذه المناطق يكون البروتين قد أنهى كامل طوله. كثير من هذه تحدث بشكل متكرر وتؤدي إلى أطوال قصيرة من البروتين، وعلى الأرجح أنها تلعب دورًا بيولوجيًا بسيطًا وسيتم استبعادهافي مزيد من التحليلات.

حسنًا، فهمت أولاً ما هي الشفرة الجينية Genetic code وكودون الحمض النووي DNA codon؟

الشفرة الجينية هو مجموعة القواعد التي تستخدمها الخلايا الحية لترجمة المعلومات المشفرة داخل المادة الجينية (تسلسل الحمض النووي أو الرنا المرسال mRNA من النوكليوتيدات الثلاثية nucleotide triplets، أو الكودونات codons) إلى بروتينات.

يتم تمثيل الشفرة الجينية القياسية تقليديًا كجدول كودون RNA لأنه عندما تصنع البروتيناتفي خلية بواسطة الريبوسومات ribosomes، فإن mRNA هو الذي يوجه عملية تخليق البروتين. يتم تحديد تسلسل mRNA بواسطة تسلسل الحمض النووي الجيني genomic DNA. فيما يلي بعض ميزات الكودونات:

1. تحدد معظم الكودونات حمضًا أمينيًا amino acid.
2. تشير ثلاثة أكواد "توقف stop " إلى نهاية البروتين.
3. يمثل كودون "البداية start "، AUG، بداية البروتين ويرمز أيضًا إلى الحمض الأميني ميثيونين amino acid methionine.



سلسلة من الكودوناتفي جزء من جزيء (mRNA). يتكون كل كودون من ثلاثة نيوكليوتيدات، تتوافق عادة مع حمض أميني واحد. يتم اختصار النيوكليوتيدات بالأحرف A و U و G و C. وهذا هو mRNA الذي يستخدم U(يوراسيل) يستخدم الحمض النووي T(الثايمين) بدلاً من ذلك. سيرشد جزيء mRNA هذا الريبوسوم لتخليق بروتين وفقًا لهذه الشفرة.

```
from Bio.Data import CodonTable
print(CodonTable.unambiguous_rna_by_name['Standard'])
```

```
Table 1 Standard, SGC0

  | U      | C      | A      | G      |
--+--------+--------+--------+--------+--
U | UUU F  | UCU S  | UAU Y  | UGU C  | U
U | UUC F  | UCC S  | UAC Y  | UGC C  | C
U | UUA L  | UCA S  | UAA Stop| UGA Stop| A
U | UUG L(s)| UCG S  | UAG Stop| UGG W  | G
--+--------+--------+--------+--------+--
C | CUU L  | CCU P  | CAU H  | CGU R  | U
C | CUC L  | CCC P  | CAC H  | CGC R  | C
C | CUA L  | CCA P  | CAA Q  | CGA R  | A
C | CUG L(s)| CCG P  | CAG Q  | CGG R  | G
--+--------+--------+--------+--------+--
A | AUU I  | ACU T  | AAU N  | AGU S  | U
A | AUC I  | ACC T  | AAC N  | AGC S  | C
A | AUA I  | ACA T  | AAA K  | AGA R  | A
A | AUG M(s)| ACG T  | AAG K  | AGG R  | G
--+--------+--------+--------+--------+--
G | GUU V  | GCU A  | GAU D  | GGU G  | U
G | GUC V  | GCC A  | GAC D  | GGC G  | C
G | GUA V  | GCA A  | GAA E  | GGA G  | A
G | GUG V  | GCG A  | GAG E  | GGG G  | G
--+--------+--------+--------+--------+--
```

جدول كودون RNA

دعنا الآن نحدد جميع البروتينات (سلاسل الأحماض الأمينية chains of amino acids)، ونفصل أساسًا عند كود الإيقاف، المميز بعلامة *. ثم دعونا نزيل أي تسلسل أقل من 20 حمضًا أمينيًا، لأن هذا هو أصغر بروتين وظيفي معروف.

```
#Identify all the Proteins (chains of amino acids)
Proteins = Amino_Acid.split('*') # * is translated stop codon
df = pd.DataFrame(Proteins)
df.describe()
print('Total proteins:', len(df))def conv(item):
    return len(item)def to_str(item):
    return str(item)df['sequence_str'] = df[0].apply(to_str)
df['length'] = df[0].apply(conv)
df.rename(columns={0: "sequence"}, inplace=True)
df.head()# Take only longer than 20
functional_proteins = df.loc[df['length'] >= 20]
print('Total functional proteins:', len(functional_proteins))
functional_proteins.describe()
```

	sequence	sequence_str	length
5	(Q, D, T, S, N, S, S, I, F, C, R, L, L, T, V, ...	QDTSNSSIFCRLLTVSSVLQPIISTSRFRPGVTER	35
6	(D, G, E, P, C, P, W, F, Q, R, E, N, T, R, P, ...	DGEPCPWFQRENTRPTQFACFTGSRRARTWLWRLRGGGLIRGTSTS	46
9	(T, A, L, C, V, H, Q, T, F, G, C, S, N, C, T, ...	TALCVHQTFGCSNCTSWSCYG	21
12	(D, T, W, C, P, C, P, S, C, G, R, N, T, S, G, ...	DTWCPCPSCGRNTSGLPQGSSS	22
39	(H, L, Q, W, G, M, S, K, F, C, I, S, L, K, F, ...	HLQWGMSKFCISLKFHNQDYSTKG	24

تحليل البروتين باستخدام الوحدة النمطية Protparam في Biopython باستخدام ProtParam.

الأدوات المتاحة في ProtParam:

- count_amino_acids: ما عليك سوى حساب عدد المرات التي يتكرر فيها الحمض الأميني في تسلسل البروتين.
- get_amino_acids_percent: نفس طريقة إرجاع الرقم بالنسبة المئوية للتسلسل بأكمله.
- molecular_weight: لحساب الوزن الجزيئي للبروتين.
- aromaticity: تحسب القيمة العطرية للبروتين.
- flexibility: تنفيذ طريقة المرونة.
- isoelectric_point: تستخدم هذه الطريقة الوحدة النمطية IsoelectricPoint لحساب pI للبروتين.
- Secondary_structure_fraction: تُرجع هذه الطريقة قائمة بكسر الأحماض الأمينية التي تميل إلى أن تكون في الحلزون helix أو الدوران turn أو الورقة sheet.
- الأحماض الأمينية في الحلزون Amino acids in Helix: V، I، Y، F، W، L.
- الأحماض الأمينية في الدور Amino acids in Turn : N، P، G، S.
- الأحماض الأمينية في الورق Amino acids in Sheet: E، M، A، L.

تحتوي القائمة على 3 قيم: [Helix، Turn، Sheet].

```
from __future__ import division
poi_list = []
from Bio.SeqUtils import ProtParam
for record in Proteins[:]:
    print("\n")
    X = ProtParam.ProteinAnalysis(str(record))
    POI = X.count_amino_acids()
```

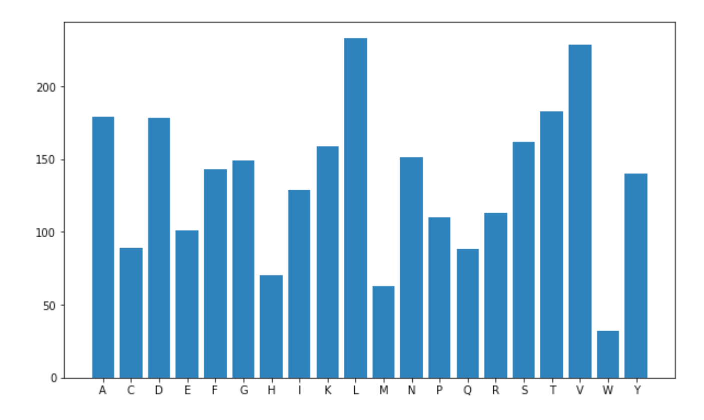
```
    poi_list.append(POI)
    MW = X.molecular_weight()
    MW_list.append(MW)
    print("Protein of Interest = ", POI)
    print("Amino acids percent =
",str(X.get_amino_acids_percent()))
    print("Molecular weight = ", MW_list)
    print("Aromaticity = ", X.aromaticity())
    print("Flexibility = ", X.flexibility())
    print("Isoelectric point = ", X.isoelectric_point())
    print("Secondary structure fraction = ",
X.secondary_structure_fraction())
```

```
Protein of Interest =  {'A': 0, 'C': 0, 'D': 0, 'E': 0, 'F': 0, 'G': 1, 'H': 0, 'I': 1, 'K':
 1, 'L': 2, 'M': 0, 'N': 0, 'P': 1, 'Q': 0, 'R': 1, 'S': 0, 'T': 0, 'V': 0, 'W': 0, 'Y': 1}
Amino acids percent =  {'A': 0.0, 'C': 0.0, 'D': 0.0, 'E': 0.0, 'F': 0.0, 'G': 0.125, 'H': 0.
0, 'I': 0.125, 'K': 0.125, 'L': 0.25, 'M': 0.0, 'N': 0.0, 'P': 0.125, 'Q': 0.0, 'R': 0.125,
 'S': 0.0, 'T': 0.0, 'V': 0.0, 'W': 0.0, 'Y': 0.125}
Molecular weight =  959.1858
Aromaticity =  0.125
Flexibility =  []
Isoelectric point =  9.99432373046875
Secondary structure fraction =  (0.5, 0.25, 0.25)

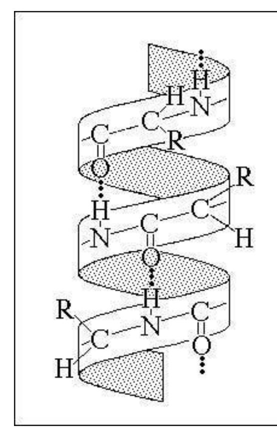

Protein of Interest =  {'A': 0, 'C': 1, 'D': 0, 'E': 0, 'F': 0, 'G': 0, 'H': 0, 'I': 1, 'K':
 0, 'L': 2, 'M': 0, 'N': 1, 'P': 0, 'Q': 2, 'R': 1, 'S': 3, 'T': 1, 'V': 1, 'W': 0, 'Y': 0}
Amino acids percent =  {'A': 0.0, 'C': 0.07692307692307693, 'D': 0.0, 'E': 0.0, 'F': 0.0, 'G':
0.0, 'H': 0.0, 'I': 0.07692307692307693, 'K': 0.0, 'L': 0.15384615384615385, 'M': 0.0, 'N': 0.
07692307692307693, 'P': 0.0, 'Q': 0.15384615384615385, 'R': 0.07692307692307693, 'S': 0.230769
23076923078, 'T': 0.07692307692307693, 'V': 0.07692307692307693, 'W': 0.0, 'Y': 0.0}
Molecular weight =  1448.6445
Aromaticity =  0.0
Flexibility =  [1.0195, 0.9710238095238095, 1.0122976190476192, 0.9595714285714285]
Isoelectric point =  8.24969482421875
Secondary structure fraction =  (0.3076923076923077, 0.3076923076923077, 0.15384615384615385)
```

ارسم النتائج:

```
MoW = pd.DataFrame(data = MW_list,columns = ["Molecular Weights"]
)#plot POI
poi_list = poi_list[48]
plt.figure(figsize=(10,6));
plt.bar(poi_list.keys(), list(poi_list.values()), align='center')
```



يبدو أن عدد Lysines (L) و Valines (V) مرتفع في هذا البروتين مما يشير إلى عدد كبير من حلزونات الفا Alpha-Helices.



الآن دعونا نقارن التشابه بين COV2 / COVID-19 و MERS و SARS.

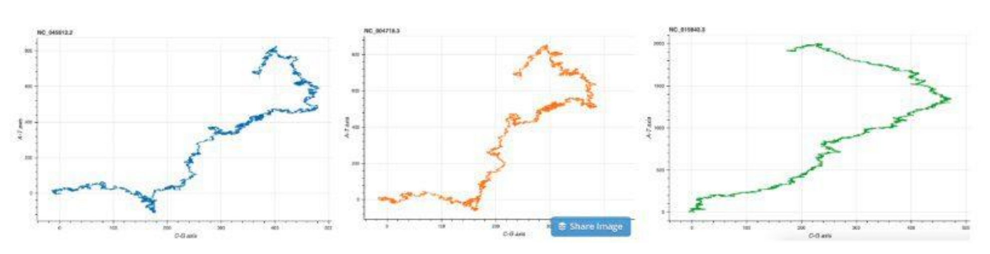
قم بتحميل ملف تسلسل الحمض النووي (FASTA) لكل من السارس وفيروس كورونا وفيروس كوفيد ـ 19.

```
#Comparing Human Coronavirus RNA
from Bio import pairwise2SARS =
SeqIO.read("/coronavirus/sars.fasta", "fasta")MERS =
SeqIO.read("/coronavirus/mers.fasta", "fasta")COV2 =
SeqIO.read("/coronavirus/cov2.fasta", "fasta")
```

أطوال التسلسل: SARS: 29751 ،COV2: 29903 ،MERS: 30119

قبل مقارنة التشابه، دعونا نتخيل الحمض النووي لكل من COV2 و SARS و MERS على التوالي.

```
#Execute on terminal
Squiggle cov2.fasta sars.fasta mers.fasta --method=gates --separate
```



تصور الحمض النووي لكل من COV2 و SARS و MERS على التوالي.

كما يمكننا أن نلاحظ أن بُنية الحمض النووي لـ COV2 و SARS متطابقة تقريبًا،في حين أن بُنية MERS مختلفة قليلاً عن الاثنين.

الآن دعونا نستخدم تقنية تراصف تسلسلي Sequence alignment لمقارنة التشابه بين جميع تسلسلات الحمض النووي.

تراصف تسلسلي هي عملية ترتيب تسلسلين أو أكثر (تسلسل DNA أو RNA أو بروتين) بترتيب معين لتحديد منطقة التشابه بينهما.

يتيح لنا تحديد المنطقة المتشابهة استنتاج الكثير من المعلومات مثل السمات المحفوظة بين الأنواع، ومدى قرب الأنواع المختلفة من الناحية الجينية، وكيف تتطور الأنواع، وما إلى ذلك.

تقارن التراصف التسلسلي الزوجي Pairwise sequence alignment تسلسلين فقط في كل مرة وتوفر أفضل محاذاة تسلسل ممكنة. من السهل فهم الأزواج واستنتاجها بشكل استثنائي من محاذاة التسلسل الناتجة.

يوفر Biopython وحدة نمطية خاصة، Bio.pairwise2 لتحديد تسلسل المحاذاة باستخدام طريقة الزوج. يطبق Biopython أفضل خوارزمية للعثور على التسلسل المتراصف وهو مكافئ للبرامج الأخرى.

```
# Alignments using pairwise2 alghoritmSARS_COV =
pairwise2.align.globalxx(SARS.seq, COV2.seq,
one_alignment_only=True, score_only=True)print('SARS/COV Similarity
(%):', SARS_COV / len(SARS.seq) * 100)MERS_COV =
pairwise2.align.globalxx(MERS.seq, COV2.seq,
one_alignment_only=True, score_only=True)print('MERS/COV Similarity
(%):', MERS_COV / len(MERS.seq) * 100)MERS_SARS =
pairwise2.align.globalxx(MERS.seq, SARS.seq,
one_alignment_only=True, score_only=True)print('MERS/SARS
Similarity (%):', MERS_SARS / len(SARS.seq) * 100)
```

```
# Alignments using pairwise2 alghoritm...

SARS/COV Similarity (%): 83.33837518066619
MERS/COV Similarity (%): 69.39141405757164
MERS/SARS Similarity (%): 69.93714496991697
```

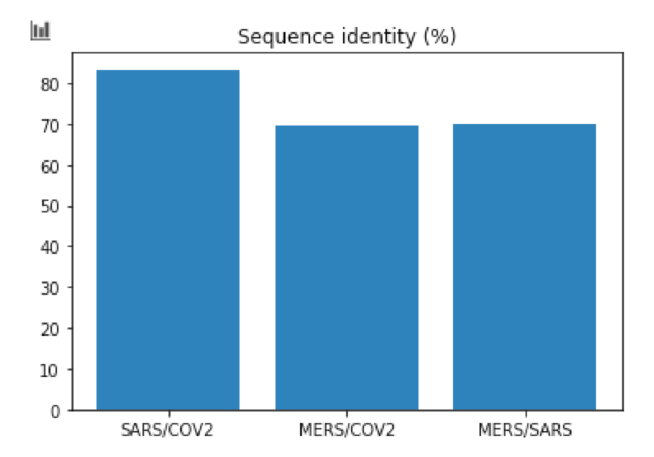
مقارنة النتائج

كشف التحليل الوراثي Phylogenetic analysis للجينوم الفيروسي الكامل (29903 نيوكليوتيدات) أن فيروس COVID-19 كان أكثر ارتباطًا (83.3 ٪ تشابه نيوكليوتيد) بمجموعة من فيروسات كورونا

الشبيهة بالسارس (جنس Betacoronavirus، subgenus Sarbecovirus) التي سبق العثور عليها في الخفافيش في الصين.

ارسم النتائج:

```
# Plot the data
X = ['SARS/COV2', 'MERS/COV2', 'MERS/SARS']
Y = [SARS_COV/ len(SARS.seq) * 100, MERS_COV/ len(MERS.seq)*100,
MERS_SARS/len(SARS.seq)*100]
plt.title('Sequence identity (%)')
plt.bar(X,Y)
```



يمكنك الحصول على الكود هنافي GitHub repo.

الاستنتاج

لذلك رأينا كيف يمكننا تفسير وتحليل بيانات تسلسل الحمض النووي لـ COVID-19، وحاولنا الحصول على العديد من الأفكار فيما يتعلق بالبروتينات التي تتكون منها.في النتيجة التي توصلنا إليها، حصلنا على عدد من Leucine (L) وValines (V) مرتفعًافي هذا البروتين مما يشير إلى وجود عدد كبير من Alpha-Helices.

في وقت لاحق قمنا بمقارنة الحمض النووي لـ COVID-19 مع MERS و SARS. ورأينا أن COVID-19 وثيق الصلة بالسارس SARS.

المصدر:

https://www.kdnuggets.com/2020/04/coronavirus-covid-19-genome-analysis-biopython.html

7) المعالجة المسبقة لتسلسل DNA المتقدم لشبكات التعلم العميق
Advanced DNA Sequences Preprocessing for Deep Learning Networks

مقدمة

تم تطبيق بنيات التعلم العميق Deep-learning مثل الشبكات العصبية العميقة Deep Neural Networks وشبكات الاعتقاد العميقة Deep Belief Networks والتعلم المعزز العميق Deep Reinforcement Learning والشبكات العصبية المتكررة Recurrent Neural Networks والشبكات العصبية التلافيفية Convolutional Neural Networks والمحولات Transformers في مجالات تشمل رؤية الكمبيوتر Computer Vision والتعرف على الكلام Speech Recognitionومعالجة اللغة الطبيعية Natural Language Processing والترجمة الآلية Machine Translation والمعلوماتية الحيوية Bioinformaticsوتصميم الأدوية Drug Design، وتحليل الصور الطبية Medical Image Analysis، وعلوم المناخ Climate Science، وفحص المواد Material Inspection، وبرامج ألعاب الطاولة Board Game programs، حيث أسفرت عن نتائج مماثلة وفي بعض الحالات تفوق أداء الخبراء البشريين.

أظهر التعلم الآلي Machine Learning (ML) نتائج واعدة جداً لتحليل تسلسل الحمض النووي DNAفي الطب السريري clinical medicine. لقد ثبت أنه يحل مهام محددة لعلم الجينوم السريري clinical genomics، والاستدعاء المتغير variant calling، والتعيين من النمط الظاهري إلى النمط الظاهري phenotype-to-phenotype mapping، وتعليقات الجينوم genome annotation، والتصنيف المتغير variant classification، إلخ.

في كثير من الحالات، عندما نطبق التعلم الآلي لمجموعات بيانات تسلسل الحمض النووي DNA sequence datasets، قد تظهر المشكلات التالية:

- لا يوجد تدريب كاف وبيانات التحقق من الصحة No enough training and validation data.
- متطلبات تنظيف البيانات وترميزها Data cleaning and encoding requirements.
- تسمية بيانات الفئات غير المتوازنة Label imbalanced classes data.

يمكن إصلاح هذه المشكلات عن طريق توليد بيانات تسلسل الحمض النووي التركيبي synthetic DNA sequence data. تقترح هذه الورقة عملية خط أنابيب بيانات Extract-Transform-Load (ETL) لحل المشكلات المذكورة أعلاه. وهي تطبق تنظيف سلسلة تسلسل الحمض النووي والتحقق من صحتها، وترميز التسميات label encoding وخوارزمية SMOTE.

مراجعة الترميز واحد ساخن لتسلسل DNA

بالنسبة لشبكات التعلم العميق Deep Learning Networks (DLN)، يجب أن يكون ترميز تسلسل الحمض النووي المطلوب هو مشفر واحد ساخن one-hot encoder كما هو موضحفي تطبيق خوارزميات التعلم الآلي لورقة تصنيف بيانات الجينوم. دعونا نحاول مراجعة وتحليل مشفر واحد ساخن لتسلسل الحمض النووي الذي يحتوي على أربعة نيوكليوتيدات رئيسية A و C و G و T. وسيتم استخدام دالة dna_sequence_one_hot_encoder(). تم تطوير هذه الدالة باستخدام كائن فئة OneHotEncoder () من scikit-Learning.

```
dna_sequence = "ACGT"
dna_onehot_encoder = PyDNA.dna_sequence_one_hot_encoder(dna_sequence)
```

النتائج:

```
DNA sequence string:
ACGT
DNA sequence list:
['A', 'C', 'G', 'T']
DNA sequence label encoder:
[0 1 2 3]
DNA sequence label encoder reshape:
[[0 1 2 3]]
DNA sequence one-hot encoder:
[[1. 0. 0. 0.]
[0. 1. 0. 0.]
[0. 0. 1. 0.]
[0. 0. 0. 1.]]
DNA sequence one-hot encoder shape:
(4, 4)
Number of rows:
4
Number of columns:
4
```

كما ترى فإن عدد الصفوف يمثل حجم (طول length) تسلسل الحمض النووي وعدد الأعمدة التي تحدد عدد النيوكليوتيدات الأساسية الفريدة. يجب أن يكون عدد الأعمدة أربعة. هذا مطلب لتطوير نماذج شبكة تعلم عميق DLN باستخدام مشفر واحد ساخن لتسلسل الحمض النووي.

دعونا نجرب تسلسل الحمض النووي بحجم 20 نيوكليوتيد قاعدي.

```
dna_sequence = "ATGATCGCATAGATGACTAG"
dna_onehot_encoder = PyDNA.dna_sequence_one_hot_encoder(dna_sequence)
```

النتائج:

```
DNA sequence string:
ATGATCGCATAGATGACTAG
DNA sequence list:
['A', 'T', 'G', 'A', 'T', 'C', 'G', 'C', 'A', 'T', 'A', 'G', 'A',
'T', 'G',
'A', 'C', 'T', 'A', 'G']
DNA sequence label encoder:
[0 3 2 0 3 1 2 1 0 3 0 2 0 3 2 0 1 3 0 2]
DNA sequence label encoder reshape:
[[0 3 2 0 3 1 2 1 0 3 0 2 0 3 2 0 1 3 0 2]]
DNA sequence one-hot encoder:
[[1. 0. 0. 0.]
 [0. 0. 0. 1.]
 [0. 0. 1. 0.]
 [1. 0. 0. 0.]
 [0. 0. 0. 1.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 1. 0. 0.]
 [1. 0. 0. 0.]
 [0. 0. 0. 1.]
 [1. 0. 0. 0.]
 [0. 0. 1. 0.]
 [1. 0. 0. 0.]
 [0. 0. 0. 1.]
 [0. 0. 1. 0.]
 [1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 0. 1.]
 [1. 0. 0. 0.]
 [0. 0. 1. 0.]]
DNA sequence one-hot encoder shape:
(20, 4)
Number of rows:
20
Number of columns:
4
```

نحصل على 20 صفًا و 4 أعمدة كما ينبغي.

في تسلسل الحمض النووي أدناه، يكون نوكليوتيد القاعدة G مفقودًا.

```
DNA sequence list:
['A', 'T', 'C', 'A', 'T', 'C', 'C', 'C', 'A', 'T', 'A', 'C', 'A',
'T', 'C',
'A', 'C', 'T', 'A', 'C']
DNA sequence label encoder:
[0 2 1 0 2 1 1 1 0 2 0 1 0 2 1 0 1 2 0 1]
DNA sequence label encoder reshape:
[[0 2 1 0 2 1 1 1 0 2 0 1 0 2 1 0 1 2 0 1]]
DNA sequence one-hot encoder:
[[1. 0. 0.]
 [0. 0. 1.]
 [0. 1. 0.]
 [1. 0. 0.]
```

```
 [0. 0. 1.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [1. 0. 0.]
 [0. 0. 1.]
 [1. 0. 0.]
 [0. 1. 0.]
 [1. 0. 0.]
 [0. 0. 1.]
 [0. 1. 0.]
 [1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]
 [1. 0. 0.]
 [0. 1. 0.]]
DNA sequence one-hot encoder shape:
(20, 3)
Number of rows:
20
Number of columns:
3
```

كما ترى، انخفض عدد الأعمدة إلى ثلاثة.في مجموعة بيانات تسلسل الحمض النووي، لن يعمل تسلسل الصف المحدد هذا مع ترميز واحد ساخن لخوارزميات التعلم الالي. على وجه التحديد، بالنسبة لـ DLN، يجب توجيه البيانات لأداء عمليات المصفوفة المطلوبة بكفاءة. هذا يعني أن عدد الصفوف والأعمدة لكل تسلسل DNA يجب أن يكون هو نفسه لمجموعة البيانات بأكملها. على سبيل المثال: إذا كان عدد الأعمدة مختلفاً، فسيتم فقد نيوكليوتيد قاعدة الحمض النووي ويحدث الخطأ التالي عند إنشاء مصفوفة ترميز الإدخال النهائي ذات الواحد الساخن.

[File Name]: \anaconda3\envs\python3.10.9\lib\site-packages\numpy\core\shape_base.py [Procedure Name]: stack [Error Message]: all input arrays must have the same shape [Error Type]: <class 'ValueError'> [Line Number]: 426 [Line Code]: raise ValueError('all input arrays must have the same shape')

بالنسبة لحالات الاستخدام هذه، يمكننا إزالة الصف (فقدان البيانات loose the data) أو استخدام نوع من حشو التسلسل sequence padding. بالنسبة لأي من هذه الحلول، يجب اختبار نموذج (نماذج) التعلم الالي المحدد بعناية باستخدام مجموعات بيانات إنتاج حقيقية.

حشوة ترميز واحد ساخن لتسلسل DNA

دعونا نلقي نظرة على حلول حشو تسلسل الحمض النووي DNA sequence padding. توفر المدونة "معالجة مدخلات الحمض النووي ذات الحجم المتغير" أساسيات حشو التسلسل باستخدام مكتبة كيراس Keras (https://keras.io/). تمت تغطية مقدمة جيدة لدالة pad_sequences() في ورقة "pad_sequences".

افترض أن لدينا تسلسلين من الحمض النووي بأحجام مختلفة. هذا هو أول واحد.

```
1. DNA sequence string:
ATGATCGCATAGATGACTAGT
DNA sequence list:
['A', 'T', 'G', 'A', 'T', 'C', 'G', 'C', 'A', 'T', 'A', 'G', 'A',
'T', 'G',
'A', 'C', 'T', 'A', 'G', 'T']
DNA sequence label encoder:
[0 3 2 0 3 1 2 1 0 3 0 2 0 3 2 0 1 3 0 2 3]
DNA sequence label encoder reshape:
[[0 3 2 0 3 1 2 1 0 3 0 2 0 3 2 0 1 3 0 2 3]]
DNA sequence one-hot encoder (dna_one_hot_encoder1):
[[1. 0. 0. 0.]
 [0. 0. 0. 1.]
 [0. 0. 1. 0.]
 [1. 0. 0. 0.]
 [0. 0. 0. 1.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 1. 0. 0.]
 [1. 0. 0. 0.]
 [0. 0. 0. 1.]
 [1. 0. 0. 0.]
 [0. 0. 1. 0.]
 [1. 0. 0. 0.]
 [0. 0. 0. 1.]
 [0. 0. 1. 0.]
 [1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 0. 1.]
 [1. 0. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
DNA sequence one-hot encoder shape:
(21, 4)
Number of rows:
21
Number of columns:
4
```

والتسلسل الثاني.

```
2. DNA sequence string:
ATGATCGCATAGATGACTAGTAGAT
DNA sequence list:
['A', 'T', 'G', 'A', 'T', 'C', 'G', 'C', 'A', 'T', 'A', 'G', 'A',
'T', 'G',
'A', 'C', 'T', 'A', 'G', 'T', 'A', 'G', 'A', 'T']
DNA sequence label encoder:
[0 3 2 0 3 1 2 1 0 3 0 2 0 3 2 0 1 3 0 2 3 0 2 0 3]
DNA sequence label encoder reshape:
[[0 3 2 0 3 1 2 1 0 3 0 2 0 3 2 0 1 3 0 2 3 0 2 0 3]]
DNA sequence one-hot encoder (dna_one_hot_encoder2):
[[1. 0. 0. 0.]
```

```
 [0. 0. 0. 1.]
 [0. 0. 1. 0.]
 [1. 0. 0. 0.]
 [0. 0. 0. 1.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 1. 0. 0.]
 [1. 0. 0. 0.]
 [0. 0. 0. 1.]
 [1. 0. 0. 0.]
 [0. 0. 1. 0.]
 [1. 0. 0. 0.]
 [0. 0. 0. 1.]
 [0. 0. 1. 0.]
 [1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 0. 1.]
 [1. 0. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]
 [1. 0. 0. 0.]
 [0. 0. 1. 0.]
 [1. 0. 0. 0.]
 [0. 0. 0. 1.]]
DNA sequence one-hot encoder shape:
(25, 4)
Number of rows:
25
Number of columns:
4
```

تسلسل الحمض النووي الأول له شكل (21, 4) والثاني (25, 4). سنحتاج إلى استخدام نوع من المساحة المتروكة لأول واحد لإضافة أربعة صفوف أخرى إليه. للقيام بذلك، سيتم استخدام دالة كيراس pad_sequences() الموضحة أدناه.

```
@staticmethod
def dna_onehot_encoder_padding(dna_onehot_encoded_list, data_type,
padding_type, padding_value):
    try:
        dna_padding =
tf.keras.preprocessing.sequence.pad_sequences(sequences=dna_onehot_e
ncoded_list,
        dtype=data_type, padding=padding_type, value=padding_value)
    except:
        print(PyDNA.get_exception_info())
        if PyDNA._app_is_log: PyDNA.write_log_file("error",
PyDNA.get_exception_info())
    return dna_padding
```

في البرنامج أدناه، يتم تمرير قائمة تسلسل الحمض النووي المشفر واحد ساخن إلى PyDNA.dna_onehot_encoder_padding() لتعبئة الأول.

```
dna_onehot_encoded_list = [dna_one_hot_encoder1,
dna_one_hot_encoder2]
```

```
for item  in dna_onehot_encoded_list:
    print("DNA one-hot encoder:\n{}".format(item))
dna_padding =
PyDNA.dna_onehot_encoder_padding(dna_onehot_encoded_list, "float32",
"pre", 0)
for padded  in dna_padding:
    print(padded.shape)
    print("DNA one-hot encoder padded:\n{}".format(padded))
```

دعونا نلقي نظرة على نتائج البرنامج لأول مشفر تسلسل الحمض النووي dna_one_hot_encoder1 قبل الحشو.

```
DNA one-hot encoder:
[[1. 0. 0. 0.]
 [0. 0. 0. 1.]
 [0. 0. 1. 0.]
 [1. 0. 0. 0.]
 [0. 0. 0. 1.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 1. 0. 0.]
 [1. 0. 0. 0.]
 [0. 0. 0. 1.]
 [1. 0. 0. 0.]
 [0. 0. 1. 0.]
 [1. 0. 0. 0.]
 [0. 0. 0. 1.]
 [0. 0. 1. 0.]
 [1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 0. 1.]
 [1. 0. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
shape: (21, 4)
```

ها هي النتائج بعد الحشو.

```
DNA one-hot encoder padded:
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [1. 0. 0. 0.]
 [0. 0. 0. 1.]
 [0. 0. 1. 0.]
 [1. 0. 0. 0.]
 [0. 0. 0. 1.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 1. 0. 0.]
 [1. 0. 0. 0.]
 [0. 0. 0. 1.]
 [1. 0. 0. 0.]
 [0. 0. 1. 0.]
```

```
 [1. 0. 0. 0.]
 [0. 0. 0. 1.]
 [0. 0. 1. 0.]
 [1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 0. 1.]
 [1. 0. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
shape: (25, 4)
```

كما ترى، تمت إضافة الصفوف الأربعة التالية ذات القيم الصفرية إلى بداية مشفر numpy غير المترابط.

```
[0. 0. 0. 0.]
[0. 0. 0. 0.]
[0. 0. 0. 0.]
[0. 0. 0. 0.]
```

إذا تغيرت قيمة الحشو إلى 1، ستكون الصفوف الأربعة الأولى كما يلي:

```
[1. 1. 1. 1.]
[1. 1. 1. 1.]
[1. 1. 1. 1.]
[1. 1. 1. 1.]
```

من المهم الإشارة إلى أنهفي حالة احتواء أي من مجموعات البيانات المشفرة ذات الترميز الواحد ساخن لتسلسل الحمض النووي على عدد مختلف من الأعمدة، فلن تعمل دالة حشو التسلسل هذه. لهذه الحالة سيحدث الخطأ التالي.

[File Name]: C:\Users\portland\anaconda3\envs\python3.10.9\lib\site-packages\keras\utils\data_utils.py [Procedure Name]: pad_sequences [Error Message]: Shape of sample (3,) of sequence at position 1 is different from expected shape (4,) [Error Type]: <class 'ValueError'> [Line Number]: 1084 [Line Code]: raise ValueError(Shape of sample (3,) of sequence at position 1 is different from expected shape (4,))

الآن بعد أن أصبح لجميع تسلسلات الحمض النووي ذات الترميز الواحد ساخن لها نفس الشكل، يمكن تطبيق أي خوارزميات DLN لتطوير النماذج التنبؤية النهائية. يجب اختبار هذا النموذج مع بيانات الإنتاج المعروفة الحقيقية.

تنظيف بيانات تسلسل DNA

أثناء تطوير التعلم الآلي، يكون تنظيف البيانات data cleaning (المعالجة المسبقة preprocessing) حوالي 60٪ ـ 70٪ من العمل بأكمله. يعرف مهندسو البيانات ذلك جيداً. بالنسبة لمشروعات تسلسل DNA للتعلم الالي باستخدام التصنيف classification والانحدار regression والتجميع clustering، تعد عملية تنظيف البيانات مهمة جداً ومطلوبة. للتحقق من تسلسل الحمض النووي، هناك متطلبان رئيسيان:

1. يجب أن تحتوي سلسلة تسلسل الحمض النووي فقط على أربعة أحرف نيوكليوتيد رئيسية A و C و G و T. أي حرف آخرفي سلسلة التسلسل سينشئ عمودًا جديداً عند تحويله إلى مشفر واحد ساخن.
2. يجب أن تحتوي سلسلة تسلسل الحمض النووي على جميع النيوكليوتيدات الأربعة الرئيسية حرف A و C و G و T. أي حرف مفقودفي سلسلة التسلسل سيقلل من كمية الأعمدة عند تحويلها إلى مشفر واحد ساخن.

دعونا نلقي نظرة على بعض حالات الاستخدام للتحقق من هذين المطلبين.في نتائج الشفرة أدناه، يمكننا أن نرى تسلسل DNA صالحًا يحتوي على جميع النيوكليوتيدات الأربعة الرئيسية. يتكون شكل الترميز الواحد الساخن من 60 صفًا و 4 أعمدة.

```
DNA sequence string:
AACTTCTCCAACGACATCATGCTACTGCAGGTCAGGCACACTCCTGCCACTCTTGCTCTT
DNA sequence validation result:
True
DNA sequence one-hot encoder shape:
(60, 4)
```

في النتائج الثانية الموضحة أدناه، يمكننا أن نرى أن تسلسل الحمض النووي يحتوي على حرف جديد "N". بسبب ذلك فشل التحقق من الصحة validation failed. تم زيادة شكل المشفر واحد ساخن بمقدار عمود آخر.

```
DNA sequence string:
AACTTCTCCAACGACATCATGCTACTGCAGGNCAGGCACACTCCTGCCACTCTTGCTCTT
DNA sequence validation result:
False
DNA sequence one-hot encoder shape:
(60, 5)
```

في نتائج الشفرة الثالثة أدناه، يمكننا أن نرى أن تسلسل الحمض النووي يفتقد حرف النوكليوتيدات الأساسي "G". فشل التحقق من الصحة بسبب هذا. تم تقليل شكل المشفر واحد ساخن بمقدار عمود واحد.

```
DNA sequence string:
AATCTTCCCAACCCCTCTCTCTTACTTTCTAATCTATCATCTACTCATCTATCCTCACTT
DNA sequence validation result:
False
DNA sequence one-hot encoder shape:
(60, 3)
```

للحالتين الثانية والثالثة أعلاه، يجب تحديث صفوف تسلسل الحمض النووي أو إزالتها. الأمر متروك لفريق إدارة مشروع التعلم الآلي لاتخاذ هذه القرارات.

مجموعة بيانات تسلسل جينات Splice-junction

تحتوي مجموعة بيانات تسلسل الجينات splice-junction splice-junction gene sequences dataset على معلومات حول الوصلات الوراثية المأخوذة من Genbank 64.1. يوضح وصف المهمة أن الجينات تتم إزالتها أثناء عملية نسخ الحمض النووي الريبي RNA، وتسمى إنترونات introns، بينما تُستخدم المناطق لتوليد mRNA وتسمى إكسونات exons. تسمى الوصلات بينهما تقاطعات splice (splice-junctions). هذه التقاطعات هي نقاط على تسلسل الحمض النووي حيث تتم إزالة الحمض النووي "الزائد superfluous" أثناء عملية تكوين البروتين في الكائنات الحية الأعلى. هناك نوعان من وصلات الربط: تقاطع exon-intron وتقاطعات intron-exon. يتكون كل تسلسل من تسلسلات الحمض النووي في مجموعة البيانات هذه من 60 نيوكليوتيد قاعدي. ينتمي كل تسلسل DNA إلى واحدة من ثلاث فئات: "EI" (تقاطع Extron-Intron)، و "IE" (تقاطع Intron-Extron) و "N" (ليس EI أو IE). هناك 767 جينة تحمل تسمية EI و 768 بعلامة IE و 1655 بعلامة N. تتمثل مهمة مجموعة البيانات هذه في تصنيف، بالنظر إلى تسلسل الحمض النووي، الحدود بين exons (أجزاء تسلسل الحمض النووي المحتفظ بها بعد الربط splicing) والإنترونات introns (أجزاء تسلسل الحمض النووي التي يتم تقسيمها).

مشاكل الفئات غير المتوازنة في تسلسل الحمض النووي

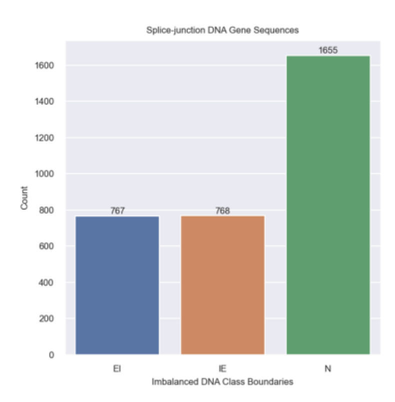
يعد التعامل مع مشكلات العالم الحقيقي لمجموعات بيانات الفئات غير المتوازنة imbalanced classes datasets باستخدام التعلم الآلي الحالي مهمة بسيطة لأنواع بيانات الميزات العددية. في الورقة البحثية "تصنيف التعبير الجيني RNA-Seq باستخدام خوارزميات التعلم الآلي"، استخدمت تقنية فرط أخذ العينات للأقليات الاصطناعية Minority Oversampling Technique (SMOTE) لمجموعة بيانات Pan-Cancer Atlas dataset RNA-Seq (HiSeq). تم تنفيذ ذلك باستخدام سطور الشجرة التالية من كود Python.

```
from imblearn.over_sampling import SMOTE
smote_over_sampling = SMOTE(random_state=50, n_jobs=-1)
X, y = smote_over_sampling.fit_resample(X, y)
```

حيث X هي مصفوفة قيم السمات features values و y هي المصفوفة (أو متجه أحادي البعد) لقيم الهدف target (التسمية label). هذا كل شيء حتى الآن، جيد وبسيط!

في مجموعات بيانات الجينوم الخاصة بنا، تكون تسلسلات الحمض النووي عبارة عن أنواع بيانات (نصية text) فئوية categorical. كما نعلم، أي خوارزميات حل الفئات غير المتوازنة تعمل مع البيانات العددية فقط. في هذه الحالة، يجب التحقق من صحة (تنظيف cleanup) مجموعة بيانات تسلسل الحمض النووي ومعالجتها مسبقاً (مشفرة encoded) من قبل. في بحثي بعنوان "تطبيق خوارزميات التعلم

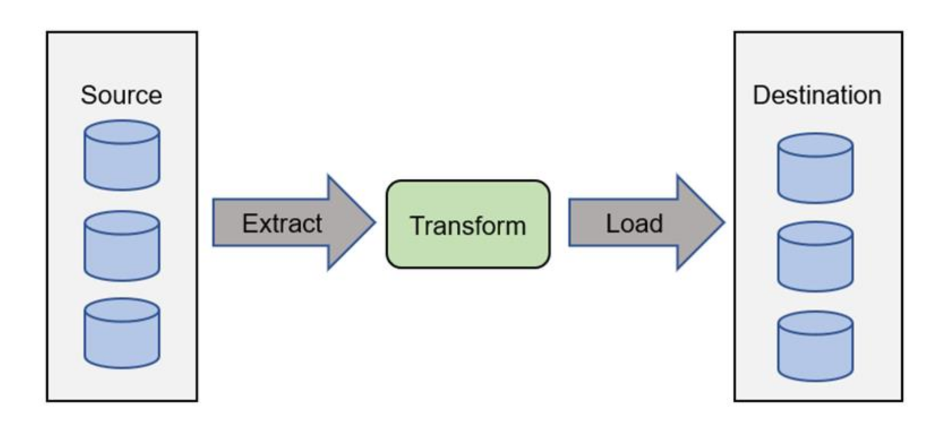
الآلي لتصنيف بيانات الجينوم"، غطيت ثلاث طرق ترميز رئيسية لسلسلة تسلسل الحمض النووي: ترميز التسمية Label Encoding، والترميز الواحد الساخن One-Hot Encoding ، وعد K-mer K-mer Counting.في هذه الورقة، سأستخدم ترميز التسمية. بالنسبة لتسلسل نيوكليوتيدات القاعدة الأربعة "ACGT" ، فإن التسمية الافتراضية المشفرة ستكون [0 ، 1 ، 2 ، 3]. بالنسبة لمجموعة البيانات الأصلية، يتم عرض مخطط حدود فئات الحمض النووي غير المتوازنة أدناه.



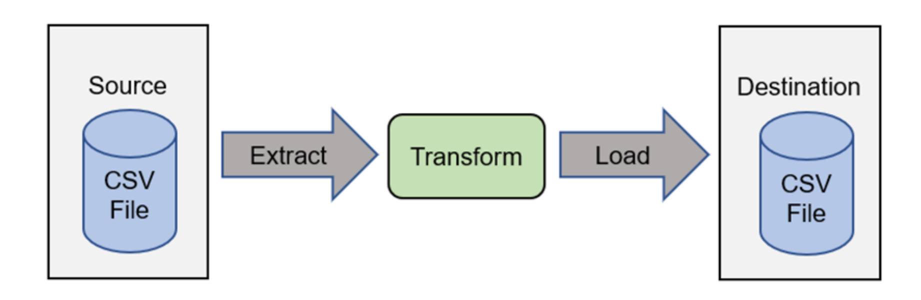
حل الفئات غير المتوازنة لتسلسل الحمض النووي المتقدم باستخدام خط أنابيب بيانات ETL

للتحقق من تسلسل الحمض النووي وتشفير مجموعة بيانات splice-junction، سأستخدم عملية أنابيب بيانات شائعة جداً تسمى إي إل تي (ETL) Extract-Transform-Load. عندما يكون "الاستخراج "Extract هو استخراج البيانات من مصدر واحد أو أكثر، فإن "التحويل "Transform هو تحويل البيانات (المعالجة المسبقة preprocessing أو التنظيف cleansing) و "التحميل "Load هو تحميل البيانات إلى وجهة واحدة أو أكثر.

"إي إل تي ETL هو الإجراء العام لنسخ البيانات من مصدر واحد أو أكثر إلى نظام الوجهة الذي يمثل البيانات بشكل مختلف عن المصدر (المصادر) أوفي سياق مختلف عن المصدر (المصادر). أصبحت عملية ETL مفهومًا شائعًافي السبعينيات وغالبًا ما تستخدمفي تخزين البيانات. يتضمن استخراج البيانات Data extraction استخراج البيانات من مصادر متجانسة homogeneous أو غير متجانسة heterogeneous؛ يعالج تحويل البيانات data transformation البيانات عن طريق تنقية البيانات وتحويلها إلى تنسيق / هيكل تخزين مناسب لأغراض الاستعلام والتحليل؛ أخيرًا، يصف تحميل البيانات data loading إدخال البياناتفي قاعدة البيانات الهدف النهائي مثل مخزن البيانات التشغيلي operational data store أو سوق البيانات data mart أو بحيرة البيانات data lake أو مستودع البيانات data warehouse ". فيما يلي رسم تخطيطي بسيط لعملية ETL.



في حالتنا، فإن المنتج النهائي لعملية ETL هذه هو إنشاء ملف CSV تم التحقق من صحته ومعالجته مسبقًا. يمكن تطبيق هذا الملف على أي خوارزميات تصنيف تحت إشراف التعلم الآلي بمافي ذلك DLN.



تم تصميم خوارزمية ETL DNA الجديدة التالية وتطويرها واختبارها باستخدام العديد من مجموعات بيانات الجينوم.

1. تنظيف مجموعة بيانات تسلسل الحمض النووي ETL

بالنسبة لعملية ETL الأولية هذه، افتح ملف مجموعة البيانات الأصلي "splice_junction_dna_sequence_original.csv" ، وقم بتطبيق تسلسل DNA الذي يقوم بمسح التحقق من الصحة وإنشاء ملف تنظيف "splice_junction_dna_sequence_cleanup.csv". فيما يلي الدالة الأولى " dna_sequence_cleanup_csv_path() ".

```
def etl_dna_sequence_cleanup(dna_sequence_original_csv_path,
dna_sequence_cleanup_csv_path, dna_sequence_columns_list):
    """splice_junction_dna_sequence original data preprocessing to
create a
        new splice_junction_dna_sequence cleanup data csv file
```

```
    args:
        dna_sequence_original_csv_path (string): dna sequence
original csv path
        dna_sequence_cleanup_csv_path (string): dna sequence cleanup
csv path
        dna_sequence_columns_list (list): list example ["dna_class",
"dna_sequence"]
    return: None
    """
    result = False
    try:
        df_genomics_original = PyDNA.pandas_read_data("CSV",
dna_sequence_original_csv_path, None)
        df_genomics_original.dropna(how="all", inplace=True)
        print("DNA sequence original data frame shape:\n{}
".format(df_genomics_original.shape))
        dna_class_list = []
        dna_sequence_list = []
        for row in df_genomics_original.itertuples():
            dna_class = row.dna_class
            dna_sequence = row.dna_sequence
            is_dna_result = PyDNA.is_dna(dna_sequence)
            if is_dna_result == True:
                dna_class_list.append(dna_class)
                dna_sequence_list.append(dna_sequence)
        df_genomics_cleanup = pd.DataFrame(list(zip(dna_class_list,
dna_sequence_list)), columns=dna_sequence_columns_list)
        print("DNA sequence cleanup data frame
shape:\n{}".format(df_genomics_cleanup.shape))
        total_remove_rows = df_genomics_original.shape[0] -
df_genomics_cleanup.shape[0]
        print("DNA sequence original total remove
rows:\n{}".format(total_remove_rows))

df_genomics_cleanup.to_csv(path_or_buf=dna_sequence_cleanup_csv_path
, index=False)
        print("DNA sequence cleanup csv file
created:\n{}".format(dna_sequence_cleanup_csv_path))
        result = True
    except:
        print(PyDNA.get_exception_info())
        if PyDNA._app_is_log: PyDNA.write_log_file("error",
PyDNA.get_exception_info())
    return result
```

تمرير المعلمات واستدعاء كود الدالة.

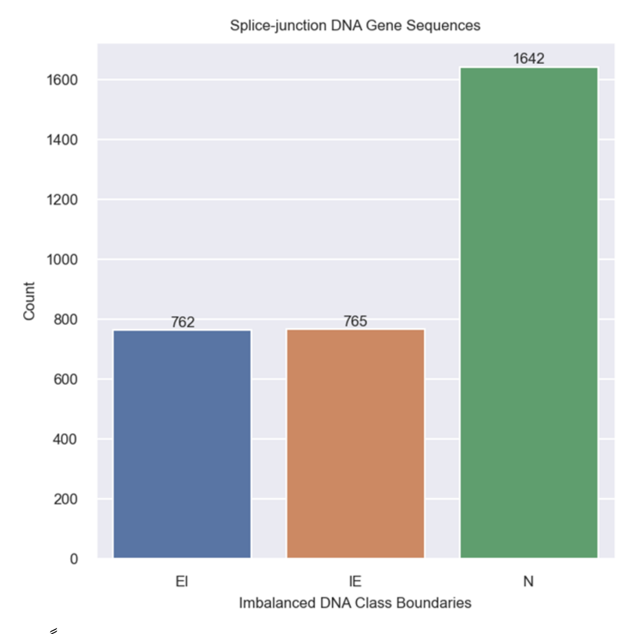
```
csv_path_folder = r"\csv_folder_path"
dna_sequence_original_csv_path = os.path.join(csv_path_folder,
"splice_junction_dna_sequence_original.csv")
dna_sequence_cleanup_csv_path = os.path.join(csv_path_folder,
```

```
"splice_junction_dna_sequence_cleanup.csv")
dna_sequence_columns_list = ["dna_class", "dna_sequence"]
result = etl_dna_sequence_cleanup(dna_sequence_original_csv_path,
dna_sequence_cleanup_csv_path, dna_sequence_columns_list)
print(result)
```

النتائج:

```
DNA sequence original data frame shape:
(3190, 2)
DNA sequence cleanup data frame shape:
(3169, 2)
DNA sequence original total remove rows:
21
DNA sequence cleanup csv file created:
\csv_folder_path\splice_junction_dna_sequence_cleanup.csv
True
```

كما ترون، فإن اول عملية ETL تزيل 21 صفاً من الملف الأصلي لتسلسل الحمض النووي. إليكم مخطط تسلسل الحمض النووي الجديد غير المتوازن لحدود الفئات.



بالنسبة إلى فئة "EI"، تمت إزالة صفوف 5، و 3 صفوف لفئة "IE" و 13 صفاً لفئة "N". أول عملية تنظيف ETL جيدة جداً.

2. ترميز تسمية مجموعة بيانات تسلسل الحمض النووي ETL

تستخدم عملية ETL الثانية الدالة " etl_dna_sequence_label_encoder() " لإنشاء الملف "splice_junction_dna_sequence_labelencoder.csv". تقوم هذه الدالة بتحويل كل صف سلسلة تسلسل DNA إلى مشفر تسمية label encoder عادي باستخدام مكتبة PyDNA. data_frame_label_encoder (dna_numpy_array) طريقة API المتوفرة في تطبيق خوارزميات التعلم الآلي لورقة مدونة تصنيف بيانات الجينوم.

تمرير المعلمات واستدعاء كود الدالة.

```
csv_path_folder = r"\csv_folder_path"
dna_sequence_cleanup_csv_path = os.path.join(csv_path_folder,
"splice_junction_dna_sequence_cleanup.csv")
dna_sequence_labelencoder_csv_path = os.path.join(csv_path_folder,
"splice_junction_dna_sequence_labelencoder.csv")
result =
etl_dna_sequence_label_encoder(dna_sequence_cleanup_csv_path,
dna_sequence_labelencoder_csv_path)
print(result)
```

النتائج:

```
DNA sequence cleanup data frame shape:
(3169, 2)
DNA sequence label encoder data frame shape:
(3169, 61)
DNA sequence label encoder csv file created:
\csv_folder_path\splice_junction_dna_sequence_labelencoder.csv
True
```

فيما يلي أول 10 صفوف من صف "EI".

```
EI,1,1,0,2,1,3,2,1,0,3,1,0,1,0,2,2,0,2,2,1,1,0,2,1,2,0,2,1,0,2,2,3,1
,3,2,3,3,1,1,0,0,2,2,2,1,1,3,3,1,2,0,2,1,1,0,2,3,1,3,2
EI,0,2,0,1,1,1,2,1,1,2,2,2,0,2,2,1,2,2,0,2,2,0,1,1,3,2,1,0,2,2,2,3,2
,0,2,1,1,1,1,0,1,1,2,1,1,1,1,3,1,1,2,3,2,1,1,1,1,1,2,1
EI,2,0,2,2,3,2,0,0,2,2,0,1,2,3,1,1,3,3,1,1,1,1,0,2,2,0,2,1,1,2,2,3,2
,0,2,0,0,2,1,2,1,0,2,3,1,2,2,2,2,2,1,0,1,2,2,2,2,0,3,2
EI,2,2,2,1,3,2,1,2,3,3,2,1,3,2,2,3,1,0,1,0,3,3,1,1,3,2,2,1,0,2,2,3,0
,3,2,2,2,2,1,2,2,2,2,1,3,3,2,1,3,1,2,2,3,3,3,3,1,1,1,1
EI,2,1,3,1,0,2,1,1,1,1,1,0,2,2,3,1,0,1,1,1,0,2,2,0,0,1,3,2,0,1,2,3,2
,0,2,3,2,3,1,1,1,1,0,3,1,1,1,2,2,1,1,1,3,3,2,0,1,1,1,3
EI,1,0,2,0,1,3,2,2,2,3,2,2,0,1,0,0,1,0,0,0,0,1,1,3,3,1,0,2,1,2,2,3,0
,0,2,0,2,0,2,2,2,1,1,0,0,2,1,3,1,0,2,0,2,0,1,1,0,1,0,2
EI,1,1,3,3,3,2,0,2,2,0,1,0,2,1,0,1,1,0,0,2,0,0,2,3,2,3,2,1,0,2,2,3,0
,1,2,3,3,1,1,1,0,1,1,3,2,1,1,1,3,2,2,3,2,2,1,1,2,1,1,0
EI,1,1,1,3,1,2,3,2,1,2,2,3,1,1,0,1,2,0,1,1,0,0,2,0,1,1,0,2,1,2,2,3,2
,0,2,1,1,0,1,2,2,2,1,0,2,2,1,1,2,2,2,2,3,1,2,3,2,2,2,2
```

```
EI,3,2,2,1,2,0,1,3,0,1,2,2,1,2,1,2,2,0,2,2,1,1,1,3,2,2,0,2,0,2,2,3,2
,0,2,2,0,1,1,1,3,1,1,3,2,3,1,1,1,3,2,1,3,1,1,0,2,3,1,1
EI,0,0,2,1,3,2,0,1,0,2,3,2,2,0,1,1,1,2,2,3,1,0,0,1,3,3,1,0,0,2,2,3,2
,0,2,1,1,0,2,2,0,2,3,1,2,2,2,3,2,2,2,0,2,2,2,3,2,0,2,0
```

3. ETL DNA Sequence Dataset SMOTE

تستخدم عملية ETL الثالثة الدالة "etl_dna_sequence_label_encoder_smote()" لإنشاء الملف "splice_junction_dna_sequence_labelencoder_smote.csv". تستخدم هذه الدالة خوارزمية تقنية زيادة عينات الأقليات الاصطناعية (SMOTE) لموازنة جميع فئات الشجرة إلى 1،642 صفًا. يجب أن يكون إجمالي الصفوف في مجموعة البيانات هذه الآن 1642 × 3 = 4926 كما ترى أدناه من حساب شكل إطار بيانات SMOTE.

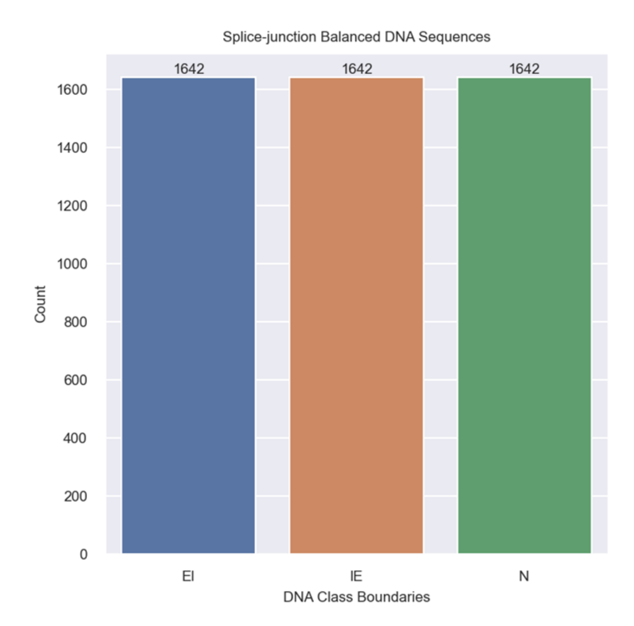
تمرير المعلمات واستدعاء كود الدالة.

```
csv_path_folder = r"\csv_folder_path"
dna_sequence_labelencoder_csv_path = os.path.join(csv_path_folder,
"splice_junction_dna_sequence_labelencoder.csv")
dna_sequence_labelencoder_smote_csv_path =
os.path.join(csv_path_folder,
"splice_junction_dna_sequence_labelencoder_smote.csv")
dna_sequence_size = 60
result =
etl_dna_sequence_label_encoder_smote(dna_sequence_labelencoder_csv_p
ath, dna_sequence_labelencoder_smote_csv_path, dna_sequence_size)
print(result)
```

النتائج:

```
DNA sequence label encoder data frame shape:
(3168, 61)
DNA sequence SMOTE data frame shape:
(4926, 61)
DNA sequence smote csv file created:
\csv_folder_path\splice_junction_dna_sequence_labelencoder_smote.csv
True
```

تم توفير طريقة PyDNA.balance_class_smote(X, y)، في مقالة (تصنيف التعبير الجيني RNA-Seq باستخدام خوارزميات التعلم الآلي). يظهر أدناه مخطط الحدود المتوازنة لفئات الحمض النووي النهائية.



4. مجموعة بيانات تسلسل الحمض النووي ETL المعالجة مسبقًا لشبكات التعلم العميق

تستخدم عملية ETL الرابعة الدالة " etl_dna_sequence_final_dln() " لإنشاء الملف "splice_junction_dna_sequence_final_dln.csv". هذه العملية تحول مشفر التسمية تسلسل الحمض النووي مرة أخرى إلى سلسلة تسلسل الحمض النووي. تم استخدام طريقة API PyDNA.dna_labelencoder_to_sequence_string (X) لذلك.

تمرير المعلمات واستدعاء كود الدالة.

```
dna_sequence_labelencoder_smote_csv_path =
os.path.join(csv_path_folder,
"splice_junction_dna_sequence_labelencoder_smote.csv")
dna_sequence_final_to_dln_csv_path = os.path.join(csv_path_folder,
"splice_junction_dna_sequence_final_to_dln.csv")
dna_sequence_size = 60
result =
etl_dna_sequence_final_dln(dna_sequence_labelencoder_smote_csv_path,
dna_sequence_final_to_dln_csv_path, dna_sequence_size)
print(result)
```

النتائج:

```
DNA sequence SMOTE data frame shape:
(4926, 61)
DNA sequence DLN data frame shape:
(4926, 2)
```

```
DNA sequence DLN csv file created:
\csv_folder_path\splice_junction_dna_sequence_final_to_dln.csv
True
```

الآن لدينا مجموعة بيانات تسلسل DNA صالحة وكاملة، يمكننا استخدام أي من خوارزميات DLN.
فيما يلي 10 أمثلة للصفوف لفئة "EI".

```
dna_class,dna_sequence
EI,GAGGTGAAGGACGTCCTTCCCCAGGAGCCGGTGAGAAGCGCAGTCGGGGGCACGGGGATG
EI,GGGCTGCGTTGCTGGTCACATTCCTGGCAGGTATGGGGCGGGGCTTGCTCGGTTTTCCCC
EI,GCTCAGCCCCCAGGTCACCCAGGAACTGACGTGAGTGTCCCCATCCCGGCCCTTGACCCT
EI,CAGACTGGGTGGACAACAAAACCTTCAGCGGTAAGAGAGGGCCAAGCTCAGAGACCACAG
EI,CCTTTGAGGACAGCACCAAGAAGTGTGCAGGTACGTTCCCACCTGCCCTGGTGGCCGCCA
EI,CCCTCGTGCGGTCCACGACCAAGACCAGCGGTGAGCCACGGGCAGGCCGGGGTCGTGGGG
EI,TGGCGACTACGGCGCGGAGGCCCTGGAGAGGTGAGGACCCTCCTGTCCCTGCTCCAGTCC
EI,AAGCTGACAGTGGACCCGGTCAACTTCAAGGTGAGCCAGGAGTCGGGTGGGAGGGTGAGA
EI,TGGCGACTACGGCGCGGAGGCCCTGGAGAGGTGAGGACCCTGGTATCCCTGCTGCCAGTC
EI,AAGCTGAGAGTGGACCCTGTCAACTTCAAGGTGAGCCACCAGTCGGGTGGGGAGGGTGAG
```

5. تنظيف مجموعة البيانات النهائية لتسلسل الحمض النووي ETL

أوصي بتطبيق أول عملية ETL لتسلسل الحمض النووي مرة أخرى للتأكد من أن لدينا مجموعة بيانات نهائية صالحة.

تمرير المعلمات واستدعاء كود الدالة.

```
csv_path_folder = r"\csv_folder_path"
dna_sequence_original_csv_path = os.path.join(csv_path_folder,
"splice_junction_dna_sequence_final_dln.csv")
dna_sequence_cleanup_csv_path = os.path.join(csv_path_folder, "
splice_junction_dna_sequence_final_dln_cleanup.csv")
dna_sequence_columns_list = ["dna_class", "dna_sequence"]
result = etl_dna_sequence_cleanup(dna_sequence_original_csv_path,
dna_sequence_cleanup_csv_path, dna_sequence_columns_list)
print(result)
```

النتائج:

```
DNA sequence final DLN data frame shape:
(4926, 2)
DNA sequence final DLN cleanup data frame shape:
(4926, 2)
DNA sequence DLN csv file created:
\csv_folder_path\splice_junction_dna_sequence_final_to_dln_cleanup.c
sv
True
```

معالجة البيانات الكاملة لتسلسل الحمض النووي ETL

يوجد أدناه كود عملية خط أنابيب بيانات تسلسل DNA ETL بالكامل.

```
# 1. ETL DNA Sequence Cleanup
csv_path_folder = r"G:\Visual
WWW\Python\1000_python_workspace\bushnell_ml_ai_project\cvs2"
dna_sequence_original_csv_path = os.path.join(csv_path_folder,
"splice_junction_dna_sequence_original.csv")
dna_sequence_cleanup_csv_path = os.path.join(csv_path_folder,
"splice_junction_dna_sequence_cleanup.csv")
dna_sequence_columns_list = ["dna_class", "dna_sequence"]
result = etl_dna_sequence_cleanup(dna_sequence_original_csv_path,
dna_sequence_cleanup_csv_path, dna_sequence_columns_list)
print("1. ETL DNA Sequence Cleanup: {}".format(result))
if result == False:
    exit()

# 2. ETL DNA Sequence Label Encoder
# csv_path_folder = r"G:\Visual
WWW\Python\1000_python_workspace\bushnell_ml_ai_project\cvs"
dna_sequence_cleanup_csv_path = os.path.join(csv_path_folder,
"splice_junction_dna_sequence_cleanup.csv")
dna_sequence_labelencoder_csv_path = os.path.join(csv_path_folder,
"splice_junction_dna_sequence_labelencoder.csv")
result =
etl_dna_sequence_label_encoder(dna_sequence_cleanup_csv_path,
dna_sequence_labelencoder_csv_path)
print("2. ETL DNA Sequence Label Encoder: {}".format(result))
if result == False:
    exit()

# 3. ETL DNA Sequence SMOTE
# csv_path_folder = r"G:\Visual
WWW\Python\1000_python_workspace\bushnell_ml_ai_project\cvs"
dna_sequence_labelencoder_csv_path = os.path.join(csv_path_folder,
"splice_junction_dna_sequence_labelencoder.csv")
dna_sequence_labelencoder_smote_csv_path =
os.path.join(csv_path_folder,
"splice_junction_dna_sequence_labelencoder_smote.csv")
dna_sequence_size = 60
result =
etl_dna_sequence_label_encoder_smote(dna_sequence_labelencoder_csv_p
ath, dna_sequence_labelencoder_smote_csv_path, dna_sequence_size)
print("3. ETL DNA Sequence SMOTE: {}".format(result))
if result == False:
    exit()

# 4. ETL DNA Sequence Final DLN
# csv_path_folder = r"G:\Visual
WWW\Python\1000_python_workspace\bushnell_ml_ai_project\cvs"
```

```
dna_sequence_labelencoder_smote_csv_path =
os.path.join(csv_path_folder,
"splice_junction_dna_sequence_labelencoder_smote.csv")
dna_sequence_final_to_dln_csv_path = os.path.join(csv_path_folder,
"splice_junction_dna_sequence_final_to_dln.csv")
dna_sequence_size = 60
result =
etl_dna_sequence_final_dln(dna_sequence_labelencoder_smote_csv_path,
dna_sequence_final_to_dln_csv_path, dna_sequence_size)
print("4. ETL DNA Sequence Final DLN: {}".format(result))
if result == False:
    exit()

# 5. ETL DNA Sequence Final DLN Cleanup
# csv_path_folder = r"G:\Visual
WWW\Python\1000_python_workspace\bushnell_ml_ai_project\cvs"
dna_sequence_final_to_dln_csv_path = os.path.join(csv_path_folder,
"splice_junction_dna_sequence_final_to_dln.csv")
dna_sequence_final_to_dln_cleanup_csv_path =
os.path.join(csv_path_folder,
"splice_junction_dna_sequence_final_to_dln_cleanup.csv")
dna_sequence_columns_list = ["dna_class", "dna_sequence"]
result =
etl_dna_sequence_cleanup(dna_sequence_final_to_dln_csv_path,
dna_sequence_final_to_dln_cleanup_csv_path,
dna_sequence_columns_list)
print("5. ETL DNA Sequence Final DLN Cleanup: {}".format(result))
if result == False:
    exit()
```

تطبيق الشبكات العصبية التلافيفية على مجموعات بيانات الفئات غير المتوازنة والمتوازنة في تسلسل الحمض النووي

في ورقة تطبيق خوارزميات التعلم الآلي لتصنيف بيانات الجينوم، تم تطبيق خوارزمية الشبكات العصبية التلافيفية (CNN) للتنبؤ بتسلسل الحمض النووي المرتبط بالبروتين. بالنسبة إلى مجموعة بيانات splice-junction لدينا، فلنكتشف كيف يعمل نموذج CNN مع الفئات غير المتوازنة والمتوازنةفي تسلسل الحمض النووي. فيما يلي نتائج الفئات غير المتوازنةفي ملف csv "splice_junction_dna_sequence_cleanup.csv".

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 49, 32)            1568
 max_pooling1d (MaxPooling1D) (None, 12, 32)              0
 flatten (Flatten)           (None, 384)                  0
 dense (Dense)               (None, 16)                6160
 dense_1 (Dense)             (None, 3)                   51
```

```
=============================================================
Total params: 7,779
Trainable params: 7,779
Non-trainable params: 0
_________________________________________________________________
Model Validation
10/10 [==============================] - 0s 2ms/step
valid accuracy score:
95.584

valid precision:
95.686

valid recall:
95.584

valid f1 score:
95.544

valid confusion matrix:
[[ 75   0   1]
 [  2  68   7]
 [  3   1 160]]

valid classification report:
                 precision    recall  f1-score   support
        0        0.94        0.99     0.96       76
        1        0.99        0.88     0.93       77
        2        0.95        0.98     0.96       164
    accuracy                          0.96       317
   macro avg     0.96        0.95     0.95       317
weighted avg     0.96        0.96     0.96       317

Model Test
10/10 [==============================] - 0s 1ms/step
test accuracy score:
95.268

test precision:
95.336

test recall:
95.268

test f1 score:
95.189

test confusion matrix:
[[ 75   0   1]
 [  4  65   7]
 [  1   2 162]]
```

```
test classification report:
                 precision    recall  f1-score   support
         0       0.94         0.99     0.96       76
         1       0.97         0.86     0.91       76
         2       0.95         0.98     0.97       165
    accuracy                           0.95       317
   macro avg     0.95         0.94     0.95       317
weighted avg     0.95         0.95     0.95       317

Model One Test
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1 entries, 0 to 0
Data columns (total 1 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   dna_sequence  1 non-null      object
dtypes: object(1)
memory usage: 136.0+ bytes
1/1 [==============================] - 0s 86ms/step
Class Predictive: 1
```

بالنسبة للفئات المتوازنة لتسلسل الحمض النووي في ملف csv "splice_junction_dna_sequence_final_to_dln_cleanup.csv" تظهر النتائج أدناه.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 49, 32)            1568
 max_pooling1d (MaxPooling1D (None, 12, 32))              0
 flatten (Flatten)           (None, 384)                  0
 dense (Dense)               (None, 16)                6160
 dense_1 (Dense)             (None, 3)                   51
=================================================================
Total params: 7,779
Trainable params: 7,779
Non-trainable params: 0
_________________________________________________________________
Model Validation
16/16 [==============================] - 0s 3ms/step
valid accuracy score:
95.732

valid precision:
95.746

valid recall:
95.732
```

```
valid f1 score:
95.733

valid confusion matrix:
[[157   3   4]
 [  3 156   5]
 [  4   2 158]]

valid classification report:
                  precision   recall  f1-score   support
           0       0.96       0.96     0.96       164
           1       0.97       0.95     0.96       164
           2       0.95       0.96     0.95       164
    accuracy                           0.96       492
   macro avg       0.96       0.96     0.96       492
weighted avg       0.96       0.96     0.96       492

Model Test
16/16 [==============================] - 0s 2ms/step
test accuracy score:
97.154

test precision:
97.19

test recall:
97.154

test f1 score:
97.16

test confusion matrix:
[[161   0   3]
 [  1 156   6]
 [  1   3 161]]

test classification report:
                  precision   recall  f1-score   support
           0       0.99       0.98     0.98       164
           1       0.98       0.96     0.97       163
           2       0.95       0.98     0.96       165
    accuracy                           0.97       492
   macro avg       0.97       0.97     0.97       492
weighted avg       0.97       0.97     0.97       492

Model One Test
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1 entries, 0 to 0
Data columns (total 1 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
```

```
 0   dna_sequence  1 non-null      object
dtypes: object(1)
memory usage: 136.0+ bytes
1/1 [==============================] - 0s 58ms/step
Class Predictive: 1
```

استنادًا إلى نتائج CNN المذكورة أعلاه، يمكننا أن نرى درجة دقة اختبار أفضل بنسبة 97.1 ٪ مع مجموعة بيانات فئات متوازنة لتسلسل الحمض النووي مقارنة بالفئات غير المتوازنة 95.2 ٪. آمل أن تفهم سبب وجوب موازنة فئات مجموعات بيانات تسلسل الحمض النووي قبل تطبيق أي خوارزميات DLN ـ إنها مهمة جدًا!في المستقبل.

الاستنتاجات

1. لتطبيق أي خوارزميات التعلم العميق على مجموعات بيانات تسلسل الحمض النووي، يلزم التحقق التالي:

- يجب أن تحتوي سلسلة تسلسل الحمض النووي على الأحرف النوكليوتيدية الأربعة الرئيسية A و C و G و T فقط. أي حرف آخرفي سلسلة التسلسل سينشئ عمودًا جديدًا عند تحويله إلى مشفر واحد ساخن.
- يجب أن تحتوي سلسلة تسلسل الحمض النووي على جميع النوكليوتيدات الأربعة الرئيسية حرف A و C و G و T. أي حرف مفقودفي سلسلة التسلسل سيقلل من كمية الأعمدة عند تحويلها إلى مشفر واحد ساخن.

2. لتوليد بيانات تسلسل الحمض النووي التركيبة، تم اقتراح عملية خط أنابيب ETL لزيادة بيانات التدريب / التحقق من الصحة وحل مشاكل الفئات غير المتوازنةفي التسمية.
3. يحسن حل مشاكل الفئات غير المتوازنة التسميةفي مجموعات بيانات تسلسل الحمض النووي أداء نماذج تصنيف شبكات التعلم العميق. يجب أن يكون هذا مطلبًا ضروريًا لأي مشروع تصنيف وانحدار وتجميع لتسلسل DNA باستخدام التعلم الآلي.

المصدر:

https://ernest-bonat.medium.com/advanced-dna-sequences-preprocessing-for-deep-learning-networks-9bf294c19d08

8) التعلم العميق على الحمض النووي القديم Deep Learning on Ancient DNA

(إعادة بناء الماضي البشري بالتعلم العميق)

الحمض النووي القديم Ancient DNA رائع! مع التطورات الحاليةفي تسلسل الجيل التالي Next Generation Sequencing (NGS)، يمكننا استخراج الحمض النووي DNA من العظام القديمة ancient bones وتسلسله وفهم الكثير عن الماضي من خلال أنواع مختلفة من التحليلات الإحصائية وعلم الوراثة السكانية Population Genetics analyses.

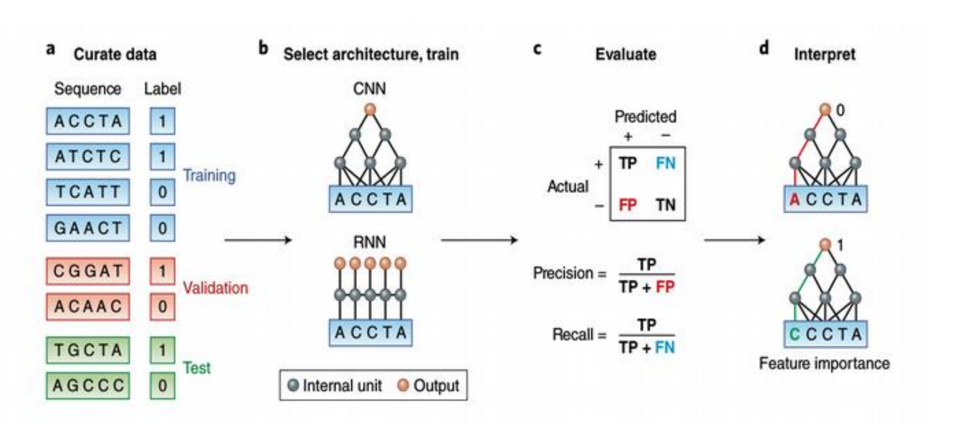
ومع ذلك، فإن التلوث الحديث modern contamination يمثل مشكلة كبيرةفي مجال أبحاث الحمض النووي القديم. اعترف Svante Pääbo، مؤسس علم الحفريات القديمة paleogenetics، أنه قامفي الغالب بتسلسل الحمض النووي الحديث modern DNA (على الأرجح ملكه)في عمله الشهير حول الحمض النووي من مومياء مصرية أدت إلى هذا المجال.في الواقع، بمجرد النظر إلى التسلسلات الحديثة والقديمة، لن تخمن أبدًا أيها جاء من الماضي. لذلك يستخدم الباحثونفي هذا المجال تحليلًا إحصائيًا متقدمًا مثل استدلال نمط ازالة الحامض الاميني deamination pattern inference، حيث تكون أداة mapDamage مفيدة جدًا. ومع ذلك، فإن مشكلة تحليل ازالة الحامض الأميني deamination analysis تكمنفي أنه يعتمد على المتوسط عبر الآلاف والملايين من التسلسلات المتوافقة، وبالتالي يكون ذلك ممكنًا فقط إذا كان لديك عينة متسلسلة بعمق (هناك حاجة إلى الكثير من الحمض النووي القديم) وجينوم مرجعي للحصول على التسلسلات تتماشى مع الجينوم الذي لا يتوفر دائمًا.

هذا هو بالضبط المكان الذي يمكننا فيه الاستفادة من قوة التعلم الآلي والعميق لاكتشاف أشكال الحمض النووي النموذجية للتسلسلات القديمة والحديثة.

التعلم العميق لعلم الجينوم

نظرًا لأن تسلسل الحمض النووي هوفي الأساس "نص بيولوجي biological text"، فيمكن تحليله باستخدام مناهج من معالجة اللغة الطبيعية Natural Language Processing أو تحليل بيانات السلاسل الزمنية Time Series data analysis. تُستخدم الشبكات العصبية الاصطناعية Artificial Neural Networks (ANNs) على نطاق واسعفي كلا المجالين وتُظهر أداءً متطورًا لعلم الجينوم Genomics أيضًا. على سبيل المثال، الشبكات العصبية التلافيفية Convolutional Neural Networks (CNNs) هي خيول عاملةفي علم الجينوم الوظيفي functional Genomics، انظر على سبيل المثال. مراجعة ممتازة لـ Argenmueller et al.، Molecular Systems Biology 12 (2016)، 878، حيث اكتشفوا بدقة عالية مواقع ربط عامل النسخ transcription factor

binding sites ومواقع التوصيل splicing sites. من أجل التنفيذ العملي لشبكات CNN لعلم الجينوم، أوصي بـ "A Primer on Deep Learning in Genomics" الرائع من Zou et al. Nature Genetics 51، pages12–18 (2019). يوضح الشكل أدناه من الورقة سير عمل نموذجي للتعليم العميق في علم الجينوم.



CNN لعلم الجينوم من Zou et al. Nature Genetics 51، pages12–18 (2019)

أعرض هنا كيفية بناء مصنف قائم على الشبكة العصبية التلافيفية (CNN) للتنبؤ لكل تسلسل للحالة القديمة لتسلسل الحمض النووي دون تعيين الجينوم المرجعي. لكن دعونا نبدأ بالنظرفي تسلسل الحمض النووي القديم.

النظر في تسلسل الحمض النووي القديم

لأغراض العرض التوضيحي، سأستخدم مسودة جينوم الإنسان البدائي draft Neanderthal genome وهو جهد تسلسل منخفض التغطية يعود إلى عام 2010. الملف عبارة عن تراصف alignment (ملف بام bam-file) لتسلسلات الحمض النووي لجينوم بشري hg18 ، ولكن لا يجب أن يكون كذلك تراصف ، يمكننا استخدام التسلسلات الأولية بشكل جيد (ملف fastq). يمكننا قراءة التسلسلاتفي Python باستخدام وحدة pysam سهلة الاستخدام ورسم توزيع أطوال التسلسلات.

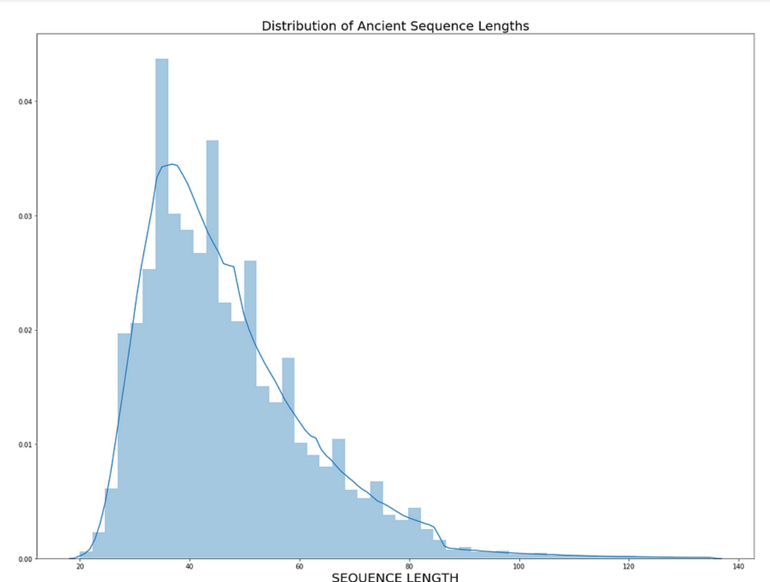
```
import os
import pysam
import numpy as np
os.chdir('/media/nikolay/My Book Duo/AncientDNA/')

# Read Bam or Fastq file (it does not have to be Bam)
neand = pysam.AlignmentFile("Neandertal.bam", "rb")

# Record length of each DNA sequence
iter = neand.fetch("chr1", 0, 249000000)
neand_lengths = []
```

```
for i in iter:
    neand_lengths.append(i.infer_query_length())

# Plot distribution of DNA read lengths
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(20,15))
sns.distplot(neand_lengths)
plt.show()
```



من الواضح أن هناك انتشارًا واسعًا لأطوال تسلسل الحمض النووي من حوالي 20 نيوكليوتيد (nt) إلى تسلسل طويل بطول 140 nt (يقرأ بلغة المعلوماتية الحيوية). هذه حقيقة معروفة تخبرنا أن الحمض النووي القديم يتحلل degraded، أي يتكسر مع الوقت إلى أجزاء صغيرة. لذلك يمكن أن يكون طول التسلسل مفيداً للغاية لاستنتاج ما إذا كان قديمًا أم لا، فكلما كان أقصر كلما زاد احتمال كونه قديمًا. ومع ذلك،في هذا التحليل، أنا مهتم بشكل خاص بنقشات motifs الحمض النووي القديمة بغض النظر عن طول التسلسل. لذلك سأقوم بتحليل التسلسلات القديمة والحديثة الطويلة على حد سواء وأترك معلومات طول التسلسل كمكافأة ممتعة يمكن للجميع استخدامها بالإضافة إلى معلومات النمط القديم. نظرًا لأنني سأستخدم عينة الحمض النووي الفرنسية الحديثة التي تم تسلسلهافي عام 2010 بطول قراءة يساوي 76 nt، سأختار فقط 76 nt من متواليات الإنسان البدائي Neanderthal طويلة ثم كمية متساوية من التسلسلات الحديثة لتدريب CNN الخاص بي.

```
# Extract 76 nt ancient DNA sequances
neand_seqs = []
for j in range(1,11):
    iter = neand.fetch('chr' + str(j))
    for i in iter:
```

```
        if i.infer_query_length() == 76:
            s = str(i.get_forward_sequence())
            if s.count('A')>0 and s.count('C')>0 and s.count('G')>0
            and s.count('T')>0 and 'N' not in s:
                neand_seqs.append(i.get_forward_sequence())

# Read modern DNA sample
modern =
pysam.AlignmentFile("/home/nikolay/WABI/Misc/AncientDNA/French.bam"
, "rb")
modern_seqs = []
for j in range(1,11):
    iter = modern.fetch("chr" + str(j))
    for i in iter:
        if len(modern_seqs) == len(neand_seqs):
            break
        else:
            s = str(i.get_forward_sequence())
            if s.count('A')>0 and s.count('C')>0 and s.count('G')>0
            and s.count('T')>0 and 'N' not in s:
                modern_seqs.append(i.get_forward_sequence())
sequences = neand_seqs + modern_seqs
labels = list(np.ones(len(neand_seqs))) +
list(np.zeros(len(modern_seqs)))
```

في المجموع، حصلنا على ما يقرب من نصف مليون قراءة، 50٪ منها قديمة. من أجل البساطة، اخترت ما يقرأ فقط من أول 10 كروموسومات. هذه الكمية من البيانات كبيرة حقًا، مقارنة بحوالي 60000 مثال من مجموعات بيانات MNIST وCIFAR10 للتعلم الآلي المستخدمة على نطاق واسع لممارسة التعرف على الصور باستخدام شبكات CNN.

تحضير بيانات الجينوم لشبكة CNN

بعد ذلك، سنقوم بإجراء قياسي لجميع خطوات CNNs أحادية الأبعاد مثل ترميز التسلسلات الواحد ساخن one-hot encoding وتقسيم البيانات إلى مجموعات فرعية للتدريب والاختبار. هنا سأتبع البروتوكول من Zou et al. *Nature Genetics* **51**, pages12–18 (2019) :

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder

# One-hot encode Sequences
integer_encoder = LabelEncoder()
one_hot_encoder = OneHotEncoder()
input_features = []
for sequence in sequences:
  integer_encoded = integer_encoder.fit_transform(list(sequence))
  integer_encoded = np.array(integer_encoded).reshape(-1, 1)
  one_hot_encoded = one_hot_encoder.fit_transform(integer_encoded)
  input_features.append(one_hot_encoded.toarray())

np.set_printoptions(threshold=40)
input_features = np.stack(input_features)
print("Example sequence\n-----------------------")
```

```
print('DNA Sequence #1:\n',sequences[0][:10],'...',sequences[0][-
10:])
print('One hot encoding of Sequence #1:\n',input_features[0].T)

# One-hot encode Labels
one_hot_encoder = OneHotEncoder()
labels = np.array(labels).reshape(-1, 1)
input_labels = one_hot_encoder.fit_transform(labels).toarray()
print('Labels:\n',labels.T)
print('One-hot encoded labels:\n',input_labels.T)

from sklearn.model_selection import train_test_split
train_features, test_features, train_labels, test_labels =
train_test_split(
    input_features, input_labels, test_size=0.25, random_state=42)
```

الآن كل شيء جاهز لتنفيذ CNN، فلنقم بذلك!

بناء وتنفيذ مصنف CNN احادي البعد

أخيرًا، حان الوقت لبدء تدريب مصنف ثنائي بسيط 1D CNN (قديم ancient مقابل غير قديم non-ancient). سوف أقوم بصنع بُنية بسيطة من كتلة واحدة تشبه VGG مع طبقتين تلافيفيتين متبوعة بطبقة Max Pooling واحدة. سأطبق أيضًا تنظيمًا صغيرًا لوزن معيار L1 لمنع الضبط الزائد overfitting.

```
from keras.optimizers import SGD, Adam, Adadelta
from keras.layers import Conv1D, Dense, MaxPooling1D, Flatten
from keras.models import Sequential
from keras.regularizers import l2, l1

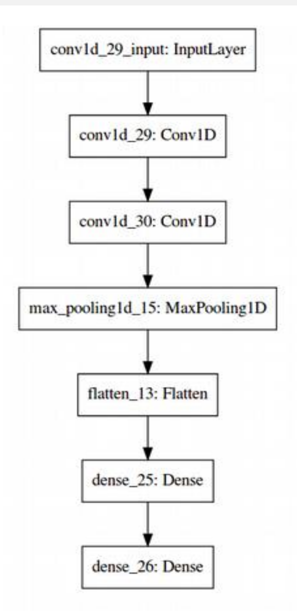
model = Sequential()

model.add(Conv1D(filters = 32, kernel_size = 5, padding = 'same',
                 kernel_initializer = 'he_uniform',
                 input_shape=(train_features.shape[1], 4),
activation = 'relu',
                 kernel_regularizer=l2(0.001)))
model.add(Conv1D(filters = 32, kernel_size = 5, padding = 'same',
                 kernel_initializer = 'he_uniform',
                 activation = 'relu',
                 kernel_regularizer=l2(0.001)))
model.add(MaxPooling1D(pool_size = 2))


model.add(Flatten())
model.add(Dense(16, kernel_initializer= 'he_uniform', activation =
'relu',
                kernel_regularizer=l2(0.001)))
model.add(Dense(2, activation='softmax'))

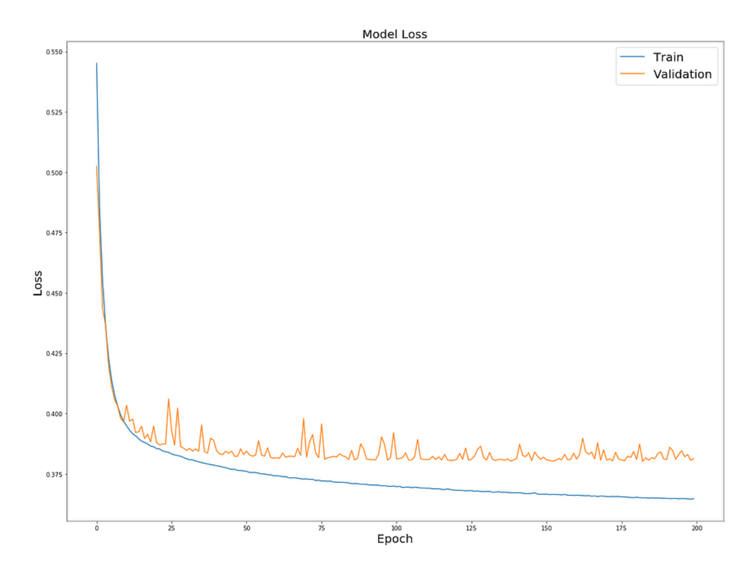
epochs = 200
lrate = 0.001
decay = lrate / epochs
```

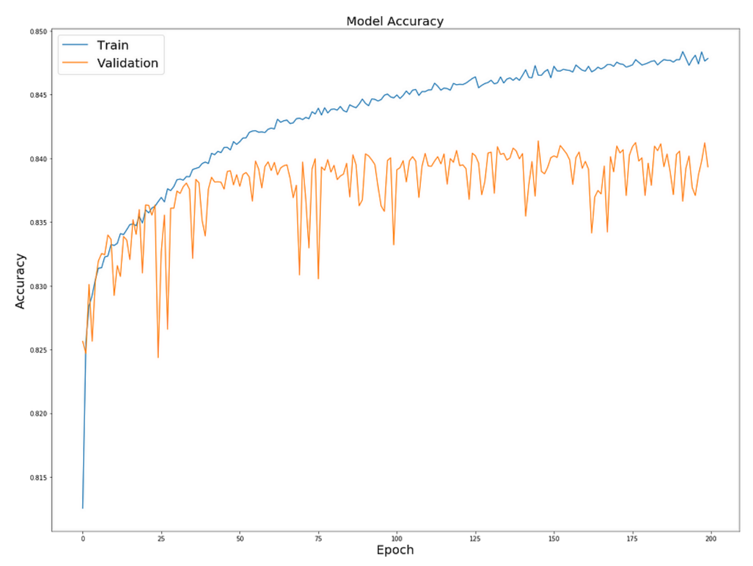
```
sgd = SGD(lr = lrate, momentum = 0.9, decay = decay, nesterov =
False)
model.compile(loss='binary_crossentropy', optimizer=sgd,
metrics=['binary_accuracy'])
model.summary()
```



بُنية تشبه VGG لمصنف 1D CNN

تم إجراء التدريب على ما يقرب من 300000 تسلسل قديم وحديث استغرق حوالي 5 ساعات على جهاز الكمبيوتر المحمول الخاص بي باستخدام 4 انوية. دعونا نتحقق من سلوك منحنيات الخطا loss والدقة accuracy:





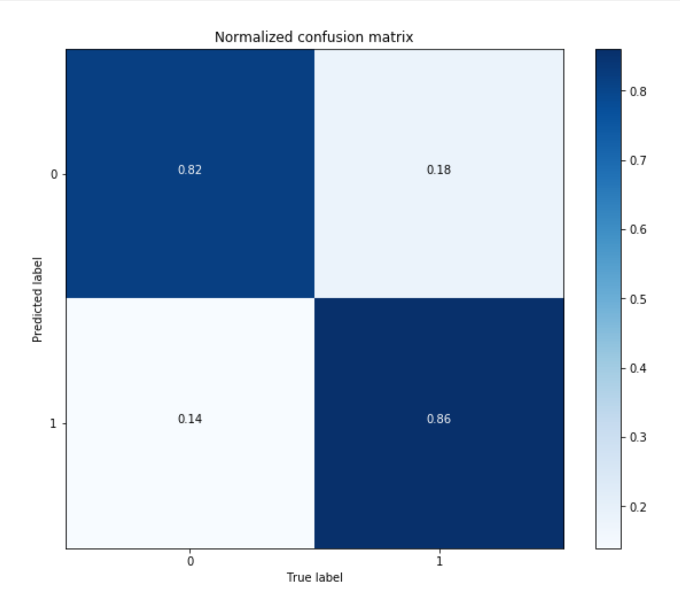
من الواضح أن النموذج غير مناسب ولكنه لا يزال يصل إلى 84 ٪ من الدقة المثيرة للإعجاب في مجموعة بيانات التحقق (حوالي 100000 تسلسل). يعتبر السلوك المتذبذب لمنحنيات التحقق من الصحة نموذجيًا لتنظيم معيار L1 ‏(L1 norm regularization). لجعلها أكثر استقرارًا، يفضل التسرب Dropout، يمكننا أيضًا زيادة حجم الدفعة batch size أو تقليل معدل التعلم learning rate. دعونا الآن نجري التقييم النهائي للنموذج على مجموعة بيانات الاختبار الثابتة مع ما يقرب من 133000 تسلسل:

```
from sklearn.metrics import confusion_matrix
import itertools

plt.figure(figsize=(10,8))
predicted_labels = model.predict(np.stack(test_features))
cm = confusion_matrix(np.argmax(test_labels, axis=1),
                      np.argmax(predicted_labels, axis=1))
print('Confusion matrix:\n',cm)

cm = cm.astype('float') / cm.sum(axis = 1)[:, np.newaxis]
plt.imshow(cm, cmap=plt.cm.Blues)
plt.title('Normalized confusion matrix')
plt.colorbar()
plt.xlabel('True label')
plt.ylabel('Predicted label')
plt.xticks([0, 1]); plt.yticks([0, 1])
plt.grid('off')
for i, j in itertools.product(range(cm.shape[0]),
range(cm.shape[1])):
    plt.text(j, i, format(cm[i, j], '.2f'),
             horizontalalignment='center',
             color='white' if cm[i, j] > 0.5 else 'black')
plt.show()
```

```
scores = model.evaluate(test_features, test_labels, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```



التقييم النهائي للنموذج يظهر مرة أخرى 84٪ من دقة التنبؤ بالحالة القديمة. مصفوفة الارتباك confusion matrix تؤكد هذا الرقم. ماذا عن تفسير CNN؟ تعد خرائط Saliency طريقة أنيقة لإثبات أهمية ميزات شبكة CNN، أي ما هي النيوكليوتيدات الأكثر إفادة للتنبؤ بالحالة القديمة لكل تسلسل:

```
import keras.backend as K

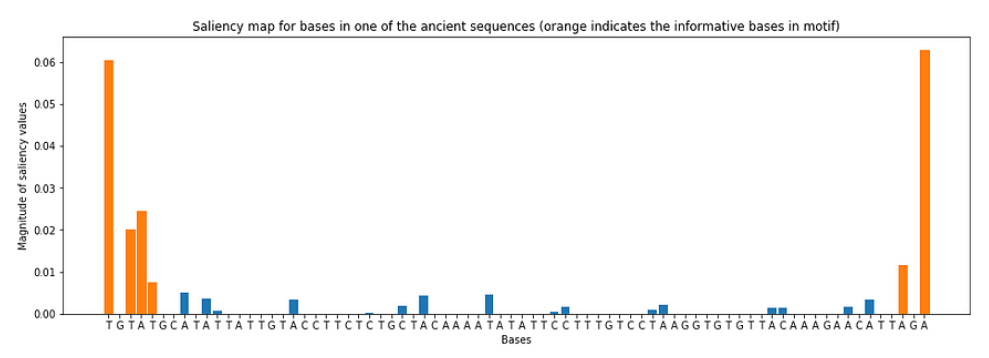
def compute_salient_bases(model, x):
    input_tensors = [model.input]
    gradients = model.optimizer.get_gradients(model.output[0][1],
model.input)
    compute_gradients = K.function(inputs = input_tensors, outputs
= gradients)

    x_value = np.expand_dims(x, axis=0)
    gradients = compute_gradients([x_value])[0][0]
    sal = np.clip(np.sum(np.multiply(gradients,x), axis=1),a_min=0,
a_max=None)
    return sal

sequence_index = 12
K.set_learning_phase(1) #set learning phase
sal = compute_salient_bases(model, input_features[sequence_index])

plt.figure(figsize=[16,5])
barlist = plt.bar(np.arange(len(sal)), sal)
[barlist[i].set_color('C1') for i in range(0,6)]
```

```
[barlist[j].set_color('C1') for j in range(71,76)]
plt.xlabel('Bases')
plt.ylabel('Magnitude of saliency values')
plt.xticks(np.arange(len(sal)), list(sequences[sequence_index]));
plt.title('Saliency map for bases in one of the ancient sequences'
          ' (orange indicates the informative bases in motif)');
plt.show()
```



نرى بوضوح أن الإشارة القديمة مقابل غير القديمة يبدو أنها تأتي من نهايات التسلسلات. هذا منطقي للغاية لأنه من المعروف أن نهايات التسلسلات القديمة تتعرض لإزالة الحامض الأميني وبالتالي تحلل مما يؤدي إلى زيادة تواتر تعدد الأشكال C / T و G / A في نهايات القراءات مقارنة بفرضية العدم null-hypothesis من التردد الموزع بشكل موحد لجميع البدائل الأخرى. هذا ما يلتقطه mapDamage. ومع ذلك، على النقيض من mapDamage، تمكنا من اكتشاف هذا النمط الخفي لإزالة الحامض الاميني دون مقارنة التسلسلات القديمة بالجينوم المرجعي. علاوة على ذلك، لدينا الآن نموذج يتنبأ بـ "القديم أو غير القديم" لكل قراءة بينما يستخدم mapDamage الاستدلال الإحصائي statistical inference من خلال حساب المتوسط عبر آلاف القراءات.

عادة، من الصعب جداً استخراج الكثير من الحمض النووي القديم، لذا فإن كل قراءة مهمة. بمعنى آخر، لا يمكننا تحمل حساب الإحصائيات عبر عدد من القراءات ولكننا نريد حالة قديمة لكل قراءة قيمة. بالإضافة إلى ذلك، غالبًا ما يتوصل تحليل الميكروبيوم القديم ancient microbiome (القراءات التي نشأت من الميكروبات القديمة بدلاً من المضيف البشري / الحيواني) إلى فيروسات قديمة تحتوي عادةً على عشرات القراءات المتوافقة مع الجينوم الفيروسي القصير جداً. في هذه الحالة، من الصعب للغاية إجراء تحليل إزالة الحامض الأميني باستخدام mapDamage بسبب نقص القوة الإحصائية lack of statistical power.

الاستنتاج

هنا أوضح أهمية التعلم العميق في مجال أبحاث الحمض النووي القديم. لقد أوضحت كيفية إنشاء مصنف بسيط للشبكة العصبية التلافيفية (CNN) للحالة القديمة مقابل الحالة غير القديمة لكل

تسلسل DNA بدون جينوم مرجعي. آمل أن تجد هذا المنشور مفيداً وأن تحصل على مصدر إلهام لتطوير التعلم العميق لمنطقة أبحاث الحمض النووي القديمة المثيرة. تحقق من نوتبوك Jupyter الكامل على github الخاص بي.

المصدر:

https://towardsdatascience.com/deep-learning-on-ancient-dna-df042dc3c73d

9) التعلم العميق لبيولوجيا الخلية المفردة Deep Learning for Single Cell Biology

(حل البُنى الخلوية عن طريق التعلم العميق)

هذه هي المقالة الثانيةفي سلسلة التعلم العميق لعلوم الحياة Deep Learning for Life Sciences. في السابق، أوضحت كيفية استخدام التعلم العميق Deep Learning على الحمض النووي القديم Ancient DNA. حان الوقت اليوم للحديث عن كيف يمكن للتعلم العميق أن يساعد بيولوجيا الخلية Cell Biologyفي التقاط تنوع diversity وتعقيد complexity مجموعات الخلايا.

أحدث تسلسل الحمض النووي الريبي أحادي الخلية Single Cell RNA sequencing (scRNAseq) ثورةفي علوم الحياة قبل بضع سنوات من خلال تقديم دقة غير مسبوقة لدراسة التباين heterogeneityفي مجموعات الخلايا. كان التأثير دراماتيكيًا لدرجة أن مجلة Science أعلنت أن تقنية scRNAseq هي الاختراق لعام 2018. كان التقدم الرئيسي هو إدراك أنه على الرغم من أن الخلايا البيولوجية قد تبدو متشابهة شكليًافي المجهر، إلا أنها يمكن أن تكون مختلفة جداً من حيث الجينات التي تعبر عنها، والتي يؤدي بدوره إلى اختلافات وظيفية بين الخلايا. لالتقاط هذا التنوع الخلوي cellular diversity، أعلن مجتمع Human Cell Atlas عن هدف طموح لبناء خريطة شاملة لتريليونات الخلايا الموجودةفي جسم الإنسان.

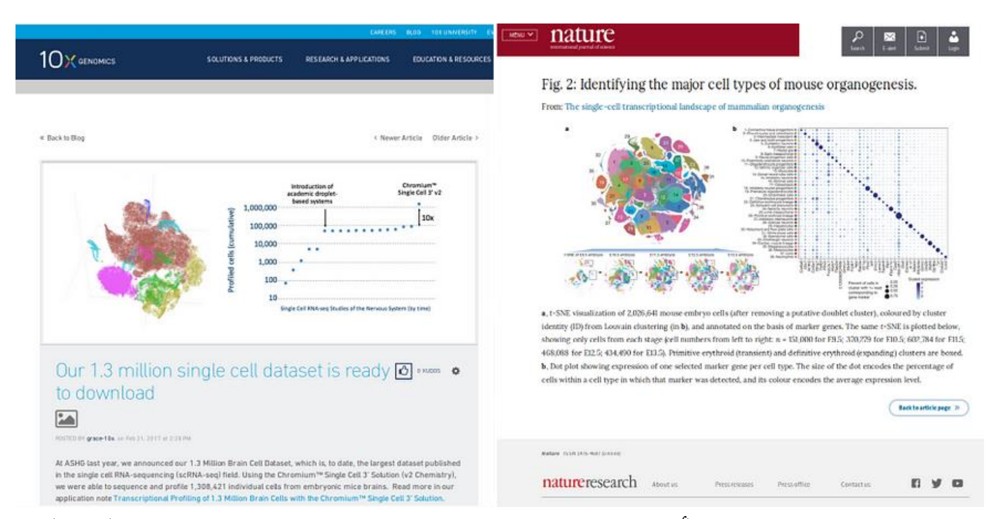
مع تطوير منصة التعبير الجيني gene expression للخلية المفردة 10X Genomics ، أصبح من المعتاد تقريبًا الحصول على معلومات كاملة عن النسخ transcriptome information من مئات الآلاف وحتى ملايين الخلايا الفردية. لذلك فإن scRNAseq (جنبًا إلى جنب مع التصوير الطبي الحيوي Biomedical Imaging وعلم الجينوم Genomics) تمثل حاليًا بيانات كبيرة حقًا تتمتع بقوة إحصائية فائقة وتفتح آفاقًا جديدة لتطبيق التعلم الآلي والعميق لتحليل بيانات الخلية المفردة Single Cell data analysis.

سأقدم هنا نظرة عامة موجزة عن الحقل، وصياغة التحديات التحليلية الرئيسية وإظهار كيفية استخدام التعلم العميق مع Keras وTensorFlow لمشاكل التعلم غير الخاضعة للإشراف unsupervised learning problemsفي تحليل بيانات تسلسل الحمض النووي الريبي أحادية الخلية single cell RNA sequencing data analysis.

لماذا تعتبر البيولوجيا الخلية المفردة مثالية للتعلم العميق؟

عند إجراء تحليل إحصائي على بعض البيانات، يتعين علينا عادةً فهم التوازن بين أ) عدد الميزات p (number of features) (الجينات genes، والبروتينات proteins، والمتغيرات الجينية genetic variants، بكسل الصورة pixel of an image وما إلى ذلك)، و ب) عدد الملاحظات

n (number of observations) (العينات samples، الخلايا cells، التسلسلات sequences إلخ.). يحتوي الجينوم البشري على ما يقرب من 20K = p من جينات ترميز البروتينفي حين أن مجموعات بيانات 10x scRNAseq المنشورة مؤخرًا تتضمن n ~ 1.3M و n ~ 2M خلايا وحيدة. هذا يتضمن أن p << n لـ scRNAseq وهو حد نموذجي للتعلم العميق، أي بالعملفي هذا الحد يمكننا تجاوز التحليل القائم على الجبر الخطي Linear Algebra والتقاط البُنية غير الخطية للغايةفي بيانات scRNAseq.

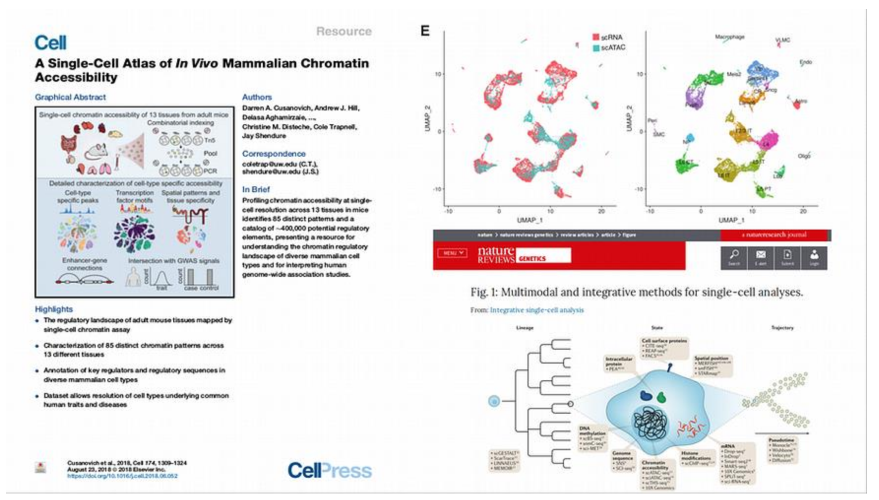


تشير الدراسات الحديثة إلى أحجام عينات غير مسبوقة من scRNAseq والتي تعد إعدادًا مثاليًا للتعلم العميق

بالنسبة للحدود الأخرى، أي عندما تكون p >> n أو p ~ n، تكون الأطر الإحصائية البايزية والمتكررة Bayesian and Frequentist statistical frameworks، على التوالي، أكثر ملاءمة. بعبارة أخرى، عند العمل مع بيانات scRNAseq، لدينا مشكلة مهمة:في حين أن الضبط الزائد overfitting وعدم القدرة على التعميم generalizability من الاهتمامات الشائعةفي علم الأحياء الحسابي Computational Biology ، هنا بالنسبة لـ scRNAseq ، علينا الانتباه إلى الضبط الناقص underfitting، أي كيفية استخدام معظم البيانات.

ليس فقط scRNAseq يزدهر حاليًافي علوم الحياة ولكن أيضًا التقنيات التي تقدم أنواعًا أخرى من المعلومات (OMICsفي مصطلحات المعلوماتية الحيوية) على مستوى خلية واحدة أصبحت أكثر شيوعًا. أدت التطورات الحديثةفي دراسة مناطق الوصول إلى الكروماتين chromatin accessibility regions (scATACseq) إلى مجموعات بيانات تحتوي على أكثر من 100 ألف خلية وحيدة.في حين أن scATACseq وحده لا يمكن أن يضمن طريقة جديدة لاكتشاف مجموعات الخلايا النادرة

rare cell populations (هدف أساسي لتحليل بيانات الخلية المفردة)، فإنه يوفر إمكانات هائلة للتكامل مع scRNAseq وبالتالي تحسين دقة تعيين الخلايا لمجموعة معينة من السكان.

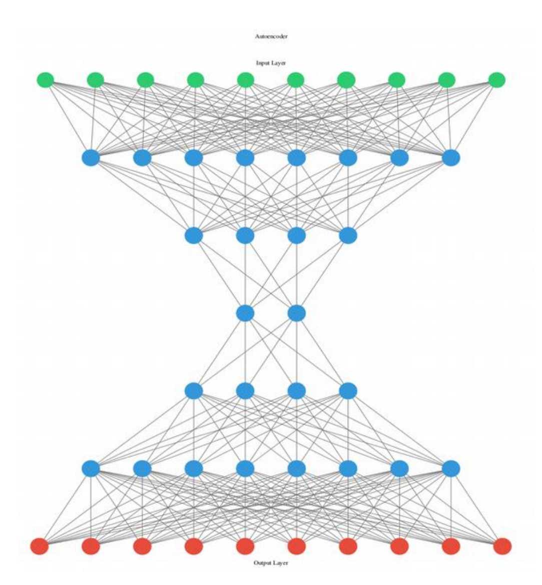


مناطق وصول الكروماتين (ATACseq) مكملة لتحليل scRNAseq

أخيرًا، لا تصل تقنيات OMICs أحادية الخلية المتعددة single cell multi-OMICs technologies (CITE-seq و scNMTseq وما إلى ذلك)، أي مصادر المعلومات المتعددة من نفس الخلايا البيولوجية، إلى أحجام عينات ضخمة نموذجية لـ scRNAseq ولكنها واعدة جداً لتحديات تكامل البيانات Data Integration المستقبلية مع التعلم العميق Deep Learning.

تقليل الأبعاد باستخدام التعلم العميق

نظرًا لأن الهدف الأساسي لتحليل scRNAseq هو اكتشاف مجموعات الخلايا الجديدة، فهو تحليل غير خاضع للإشراففي مصطلحات التعلم الآلي. ومن ثم، فإن أهم طريقتين تحليليتين مستخدمةفي scRNAseq هما تقليل الأبعاد dimensionality reduction والتكتل clustering. المشفرات التلقائية Autoencoder هي شبكة عصبية اصطناعية Artificial Neural Network (ANN) غير خاضعة للإشراف مع بنية "الفراشة butterfly " ذات المظهر المسل والتي غالبًا ما تستخدم لتقليل الأبعاد. على عكس التقنيات الخطية مثل تحليل المكونات الرئيسية Principal Component Analysis (PCA)، والتحجيم متعدد الأبعاد Multi-Dimensional Scaling (MDS)، وتحليل العامل Factor Analysis (FA) وما إلى ذلك، تقوم المشفرات التلقائية بتقليل الأبعاد غير الخطية non-linear dimensionality reduction، وبالتالي يمكنها التقاط بُنية غير خطية للغاية لبيانات الخلية المفردة.



Autoencoder هو عبارة عن شبكة عصبية اصطناعية (ANN) ببُنية "الفراشة"

سأوضح هنا الاختلاف في دقة الخلية المفردة بين تقنيات تقليل الأبعاد الخطية (PCA) وغير الخطية (Autoencoder) باستخدام مجموعة بيانات 8K ~ CITEseq scRNAseq في دم الحبل السري (CBMCs) كمثال. يرجى ملاحظة أن الكود أدناه يفترض أن ملف الإدخال بتنسيق جدولي tabular format مع وجود الجينات كأعمدة وخلايا كصفوف، ويجب أن يكون العمود الأخير من الملف عبارة عن تعليق توضيحي للخلية تم الحصول عليه باستخدام تقنية تجميع scRNAseq (scRNAseq clustering) المفضلة لديك. أوصي باستخدام التجميع المستند إلى الرسم البياني graph-based clustering مع اكتشاف مجتمع Louvaine ، وهو قوي للبيانات عالية الأبعاد، ويتم تنفيذه في سير عمل Seurat scRNAseq الشهير.

```
import numpy as np
import pandas as pd
from keras.layers import Dense
import matplotlib.pyplot as plt
from sklearn.manifold import TSNE
from keras.optimizers import Adam
from sklearn.decomposition import PCA
from keras.models import Sequential, Model

# READ AND LOG-TRANSFORM DATA
expr = pd.read_csv('MouseBrain_10X_1.3M.txt',sep='\t')
X = expr.values[:,0:(expr.shape[1]-1)]
Y = expr.values[:,expr.shape[1]-1]
X = np.log(X + 1)

# REDUCE DIMENSIONS WITH PRINCIPAL COMPONENT ANALYSIS (PCA)
n_input = 50
```

```
x_train = PCA(n_components = n_input).fit_transform(X); y_train = Y
plt.scatter(x_train[:, 0], x_train[:, 1], c = y_train, cmap =
'tab20', s = 10)
plt.title('Principal Component Analysis (PCA)')
plt.xlabel("PC1")
plt.ylabel("PC2")

# REDUCE DIMENSIONS WITH AUTOENCODER
model = Sequential()
model.add(Dense(30,       activation='elu',
input_shape=(n_input,)))
model.add(Dense(20,       activation='elu'))
model.add(Dense(10,       activation='elu'))
model.add(Dense(2,        activation='linear', name="bottleneck"))
model.add(Dense(10,       activation='elu'))
model.add(Dense(20,       activation='elu'))
model.add(Dense(30,       activation='elu'))
model.add(Dense(n_input,  activation='sigmoid'))
model.compile(loss = 'mean_squared_error', optimizer = Adam())
model.fit(x_train, x_train, batch_size = 128, epochs = 500, verbose
= 1)
encoder = Model(model.input, model.get_layer('bottleneck').output)
bottleneck_representation = encoder.predict(x_train)
plt.scatter(bottleneck_representation[:,0],
bottleneck_representation[:,1],
            c = y_train, s = 10, cmap = 'tab20')
plt.title('Autoencoder: 8 Layers')
plt.xlabel("Dimension 1")
plt.ylabel("Dimension 2")
```

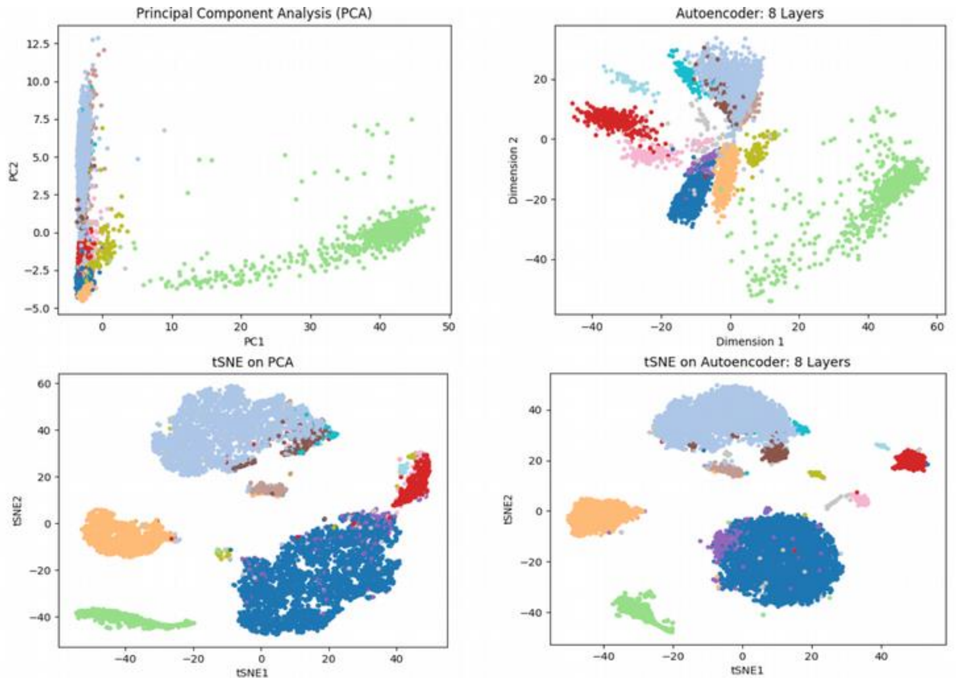
للمقارنة، سأضيف هنا مخطط تضمين Stochastic Neighbor Embedding (tSNE) الموزع على شكل حرف t، وهو عبارة عن تقنية تقليل الأبعاد غير الخطية القياسية الذهبية الحاليةفي منطقة scRNAseq. تتمثل إحدى مشكلات tSNEفي أنه لا يمكنه التعامل مع البيانات عالية الأبعاد مثل scRNAseq. لذلك، تتمثل الممارسة الشائعةفي إجراء PCA (خطي!) كتخفيض مسبق للأبعاد وتغذية الإخراجفي tSNE. ومع ذلك، يمكننا القيام بذلك بشكل أفضل من خلال إجراء خطوة تقليل الأبعاد المسبقة بطريقة غير خطية باستخدام المشفر التلقائي. دعونا نعرض مخططات tSNE لكلا الاستراتيجيتين:

```
# TSNE ON PCA
model_tsne = TSNE(learning_rate = 200, n_components = 2,
random_state = 123,
                  perplexity = 90, n_iter = 1000, verbose = 1)
tsne = model_tsne.fit_transform(x_train)
plt.scatter(tsne[:, 0], tsne[:, 1], c = y_train, cmap = 'tab20', s
= 10)
plt.title('tSNE on PCA')
plt.xlabel("tSNE1")
plt.ylabel("tSNE2")

# TSNE ON AUTOENCODER
model = Sequential()
```

```
model.add(Dense(10,      activation = 'elu',
input_shape=(X.shape[1],)))
model.add(Dense(8,       activation = 'elu'))
model.add(Dense(6,       activation = 'elu'))
model.add(Dense(4,       activation = 'linear', name =
"bottleneck"))
model.add(Dense(6,       activation = 'elu'))
model.add(Dense(8,       activation = 'elu'))
model.add(Dense(10,      activation = 'elu'))
model.add(Dense(X.shape[1],    activation = 'sigmoid'))
model.compile(loss = 'mean_squared_error', optimizer = Adam())
model.fit(X, X, batch_size = 128, epochs = 100, shuffle = True,
verbose = 1)
encoder = Model(model.input, model.get_layer('bottleneck').output)
bottleneck_representation = encoder.predict(X)

model_tsne_auto = TSNE(learning_rate = 200, n_components = 2,
random_state = 123,
                       perplexity = 90, n_iter = 1000, verbose = 1)
tsne_auto =
model_tsne_auto.fit_transform(bottleneck_representation)
plt.scatter(tsne_auto[:, 0], tsne_auto[:, 1], c = Y, cmap =
'tab20', s = 10)
plt.title('tSNE on Autoencoder: 8 Layers')
plt.xlabel("tSNE1")
plt.ylabel("tSNE2")
```



إذا لم تكن معتادًا على النظر إلى هذا النوع من المخططات، فهنا نقطة واحدة هي خلية واحدة، والألوان تتوافق مع أنواع مختلفة من الخلايا. من المفترض أن تراقب 12 خلية من الخلايا ولكن لا يمكنك

رؤيتها بشكل أساسي (تتداخل الخلايا بشكل كبير cells heavily overlap) من مخطط PCA لأن تقليل الأبعاد الخطية لا يمكن أن يحل بنية الخلية المفردة. تبدو صورة المشفر التلقائي أفضل، ويمكن اكتشاف مجموعات الخلايا المختلفة بوضوح. يوفر tSNE بشكل عام نقط خلية أكثر وضوحًا. ومع ذلك، بالنسبة لهذه الحالة بالذات، يبدو أن tSNE على المشفر التلقائي يوفر مجموعات أكثر كثافة وشفافية خاصة بالنسبة لمجموعة الخلايا الأرجواني التي تم تلطيخهافي جميع أنحاء الكتلة الزرقاءفي tSNE على مخطط PCA. وبالتالي، يعد التعلم العميق واعداً لتحسين دقة اكتشاف مجموعات الخلايا الجديدة.

البحث عن تقليل الأبعاد القابل للقياس

بالإضافة إلى صعوبة التعامل مع البيانات عالية الأبعاد، فإن مقياس tSNE سيئ عندما يصل عدد الخلايا إلى مئات الآلاف والملايين. FItSNE هو تعديل جديد واعد لـ Barnes-Hut tSNE والذي يبدو أنه يتسع بشكل أفضل لكميات كبيرة من البيانات. ومع ذلك، عند تشغيل FItSNE على 1.3 مليون خلية دماغيةفي الفأر، واجهت مشاكل في تركيبهافي الذاكرة. على وجه الخصوص، كنت أرغب في الحصول على مخططات tSNE لارتباكات perplexities عالية من أجل التحقق من الهيكل العالمي للبيانات. ومع ذلك، كان الارتباك (perplexity) = 350 هو الحد الأقصى من الارتباك الذي تمكنت من الوصول إليه باستخدام عقدة ذات ذاكرة وصول عشوائي (RAM) بسعة 256 جيجابايت على مجموعة الكمبيوتر. يعد تشغيل FItSNE أمرًا بسيطًا للغاية ويشبه الطريقة التي تؤدي بها tSNEفي R (مرة أخرى، يحتوي عمود الكتلة Cluster column على التعليقات التوضيحية للخلية):

```
library("data.table")
source("fast_tsne.R")
expr <-
suppressWarnings(as.data.frame(fread("10X_Mouse_Brain_1.3M.txt",sep
="\t")))
my_color<-as.numeric(as.character(expr$Cluster))
expr$Cluster<-NULL
expr<-as.data.frame(t(expr))
N_cells<-dim(expr)[2]
print(paste0("DATASET CONTAINS ",dim(expr)[1]," GENES AND
",N_cells," CELLS"))
tsne_opt_perp <- fftRtsne(t(log10(expr+20)),check_duplicates=FALSE,
                          perplexity=350,dims=2,max_iter=5000)
plot(tsne_opt_perp$Y,col=my_color,xlab="tSNE1",ylab="tSNE2",cex=0.5
)
```

يعد التقريب والإسقاط المتشعب الموحد Uniform Manifold Approximation and Projection (UMAP) تقنية أخرى مثيرة للاهتمام للغاية لتقليل الأبعاد غير الخطية والتي يبدو أنها تتفوق حاليًا على tSNEفي العديد من الجوانب. إنه أسرع من tSNE، بنفس سرعة FItSNE ولكنه لا يتطلب قدرًا كبيرًا من الذاكرة، ويبدو أنه يلتقط كلاً من البنية المحلية والعالمية لبيانات scRNAseq.

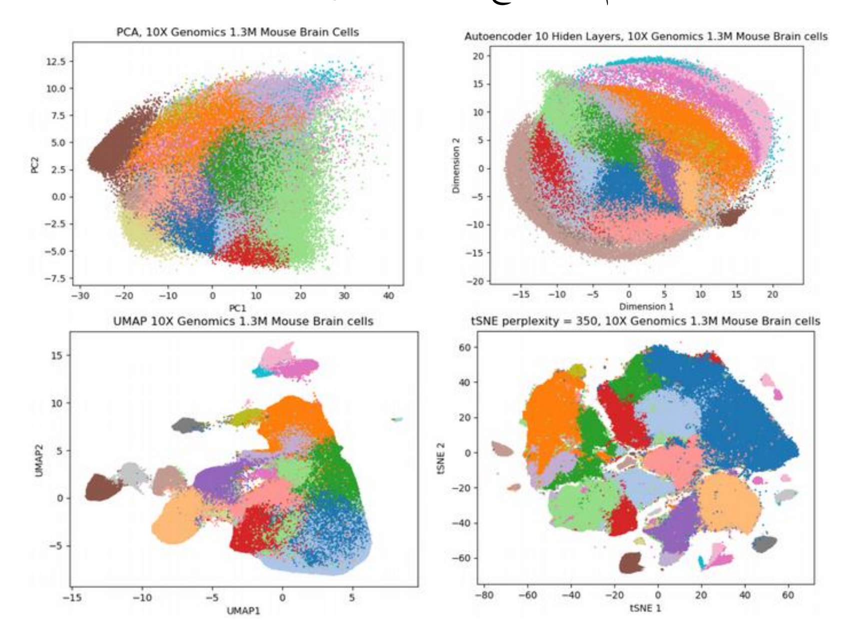
ربما يكون العيب الوحيد الذي أراه هو الرياضيات غير الشفافة nontransparent mathematics بعض الشيء وراء UMAP التي أواجهها حاليًا.

```
from umap import UMAP
model = UMAP(n_neighbors = 30, min_dist = 0.3, n_components = 2)
umap = model.fit_transform(X_reduced)
umap_coords = pd.DataFrame({'UMAP1':umap[:, 0], 'UMAP2':umap[:,
1]})
umap_coords.to_csv('umap_coords_10X_1.3M_MouseBrain.txt', sep='\t')
plt.scatter(umap[:, 0], umap[:, 1], c = Y, cmap = 'tab20', s = 1)
plt.title('UMAP')
plt.xlabel("UMAP1")
plt.ylabel("UMAP2")
```

تم مؤخرًا إصدار عدد قليل من الأساليب المثيرة للاهتمام استنادًا إلى Variational Autoencoders. أحدها هو SCVIS وهي شبكة عصبية تلتقط وتصور الهياكل منخفضة الأبعادفي بيانات التعبير الجيني للخلية الوحيدة، بالإضافة إلى أنها تحافظ على كل من هياكل الجوار المحلية والعالمية. لتشغيل SCVIS على 1.3 مليون خلية دماغيةفي الفأر، استخدمت سطر الأوامر التالي:

```
python scvis train --data_matrix_file 10X_1.3M.txt --out_dir
out_scvis
--data_label_file 10X_1.3M_MouseBrain_CellAnnotationSeurat.txt
--verbose --verbose_interval 50 --show_plot
```

أقدم أدناه مقارنة بين تقنيات تقليل الأبعاد الأربعة المذكورة، مثل PCA و FItSNE / tSNE و UMAP و SCVIS باستخدام خلايا دماغ الفأر M1.3 من 10X Genomics:



مقارنة الكروماتين لتقنيات الحد من الأبعاد على مجموعة بيانات دماغ الماوس 10X1.3 M

بالنسبة لحالة CITEseq أعلاه، يمكننا هنا أن نرى أنه على عكس PCA، فإن تقنيات تقليل الأبعاد غير الخطية (SCVIS و UMAP و tSNE / FItSNE) قادرة على حل جميع مجموعات الخلايا في بيانات scRNAseq. من بين التقنيات الثلاثة، كان UMAP هو الأسرع وقدم تمثيلًا منخفضًا جيدًا للبيانات. لكي نكون أكثر دقة حول الأوقات الحسابية، استغرق SCVIS حوالي 6 ساعات، واستغرق FItSNE 3 ساعات والكثير من الذاكرة، استغرق UMAP حوالي 3 ساعات على جهاز الكمبيوتر المحمول الخاص بي.

مع الأخذفي الاعتبار الكميات المتزايدة من بيانات scRNAseq، أتوقع أن يحل UMAP وAutoencoders محل tSNEفي المستقبل.

المشفر التلقائي العميق لـ scRNAseq مع Keras

أخيرًا، سأوضح هنا كيفية التنفيذ من البداية وتشغيل برنامج مشفر تلقائي عميق Deep Autoencoder باستخدام Keras. كما سترى، الأمر ليس بهذه الصعوبة. من أجل تجنب الصعوبات في تحميل مجموعة البيانات بأكملهافي الذاكرة، اخترت أفضل 19 مكونًا رئيسيًا ووجدت أنها مهمة من خلال إعادة التشكيل، أي خلط مصفوفة التعبير الجيني والتحقق من النسبة المئوية للتباين الموضحة بواسطة المصفوفة المضمنة permuted matrix (فرضية العدم null hypothesis). خفضت المشفر التلقائي الأبعاد تدريجياً من 19 إلى 2 (عنق الزجاجة bottleneckفي المشفر التلقائي)، حيث كانت كل طبقة مخفية تقلل بعدًا واحدًا.

```
import numpy as np
import pandas as pd
from keras.layers import Dense
import matplotlib.pyplot as plt
from keras.optimizers import Adam
from sklearn.decomposition import PCA
from keras.models import Sequential, Model
from sklearn.preprocessing import MinMaxScaler

# READ DATA AND LOG-TRANSFORM DATA
expr = pd.read_csv('MouseBrain_10X_1.3M.txt', sep = '\t', header =
None)
X = expr.values[:,0:(expr.shape[1]-1)]
Y = expr.values[:,expr.shape[1]-1]
X = np.float32( np.log(X + 1) )

# REDUCE DIMENSIONS WITH PRINCIPAL COMPONENT ANALYSIS (PCA)
n_input = 19
x_train = PCA(n_components = n_input).fit_transform(X)
y_train = Y
x_train = MinMaxScaler().fit_transform(x_train)

# REDUCE DIMENSIONS WITH AUTOENCODER
model = Sequential()
```

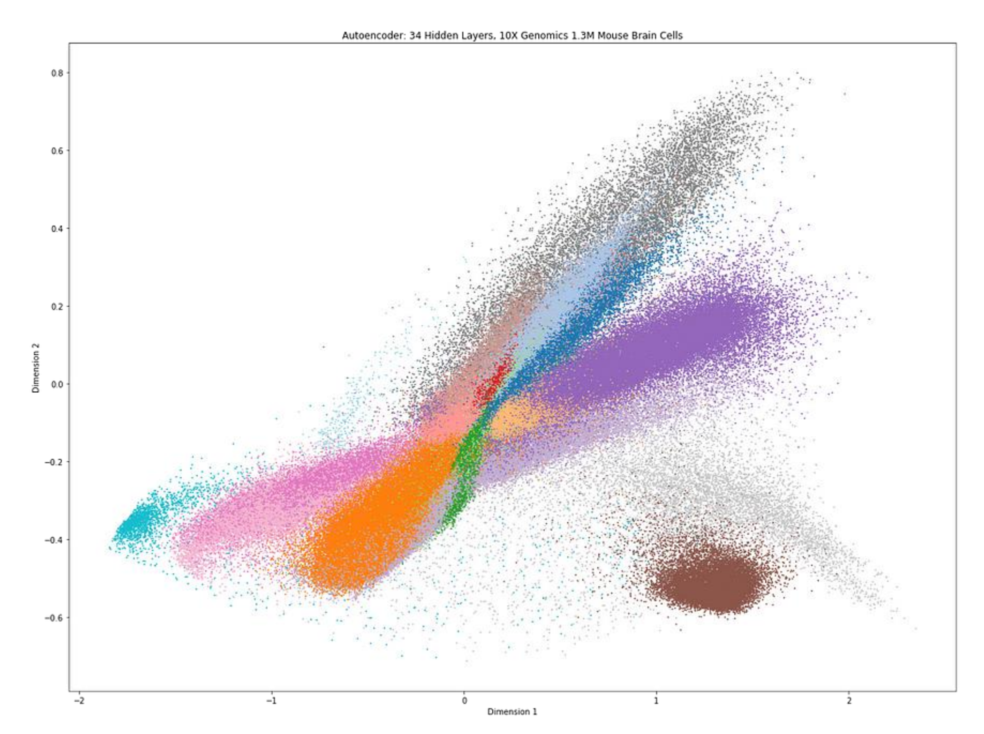
```
model.add(Dense(18,      activation='elu',
kernel_initializer='he_uniform',
                input_shape=(n_input,)))
model.add(Dense(17,      activation='elu',
kernel_initializer='he_uniform'))
model.add(Dense(16,      activation='elu',
kernel_initializer='he_uniform'))
model.add(Dense(15,      activation='elu',
kernel_initializer='he_uniform'))
model.add(Dense(14,      activation='elu',
kernel_initializer='he_uniform'))
model.add(Dense(13,      activation='elu',
kernel_initializer='he_uniform'))
model.add(Dense(12,      activation='elu',
kernel_initializer='he_uniform'))
model.add(Dense(11,      activation='elu',
kernel_initializer='he_uniform'))
model.add(Dense(10,      activation='elu',
kernel_initializer='he_uniform'))
model.add(Dense(9,       activation='elu',
kernel_initializer='he_uniform'))
model.add(Dense(8,       activation='elu',
kernel_initializer='he_uniform'))
model.add(Dense(7,       activation='elu',
kernel_initializer='he_uniform'))
model.add(Dense(6,       activation='elu',
kernel_initializer='he_uniform'))
model.add(Dense(5,       activation='elu',
kernel_initializer='he_uniform'))
model.add(Dense(4,       activation='elu',
kernel_initializer='he_uniform'))
model.add(Dense(3,       activation='elu',
kernel_initializer='he_uniform'))
model.add(Dense(2,       activation='linear',
kernel_initializer='he_uniform',
                name="bottleneck"))
model.add(Dense(3,       activation='elu',
kernel_initializer='he_uniform'))
model.add(Dense(4,       activation='elu',
kernel_initializer='he_uniform'))
model.add(Dense(5,       activation='elu',
kernel_initializer='he_uniform'))
model.add(Dense(6,       activation='elu',
kernel_initializer='he_uniform'))
model.add(Dense(7,       activation='elu',
kernel_initializer='he_uniform'))
model.add(Dense(8,       activation='elu',
kernel_initializer='he_uniform'))
model.add(Dense(9,       activation='elu',
kernel_initializer='he_uniform'))
model.add(Dense(10,      activation='elu',
kernel_initializer='he_uniform'))
model.add(Dense(11,      activation='elu',
kernel_initializer='he_uniform'))
```

```
model.add(Dense(12,      activation='elu',
kernel_initializer='he_uniform'))
model.add(Dense(13,      activation='elu',
kernel_initializer='he_uniform'))
model.add(Dense(14,      activation='elu',
kernel_initializer='he_uniform'))
model.add(Dense(15,      activation='elu',
kernel_initializer='he_uniform'))
model.add(Dense(16,      activation='elu',
kernel_initializer='he_uniform'))
model.add(Dense(17,      activation='elu',
kernel_initializer='he_uniform'))
model.add(Dense(18,      activation='elu',
kernel_initializer='he_uniform'))
model.add(Dense(n_input, activation='sigmoid'))
model.compile(loss = 'mean_squared_error', optimizer = Adam(lr =
0.0001))
model.summary()

# FIT AUTOENCODER MODEL
history = model.fit(x_train, x_train, batch_size = 4096, epochs =
100,
                    shuffle = False, verbose = 0)
print("\n" + "Training Loss: ", history.history['loss'][-1])
plt.figure(figsize=(20, 15))
plt.plot(history.history['loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.show()

encoder = Model(model.input, model.get_layer('bottleneck').output)
bottleneck_representation = encoder.predict(x_train)

# PLOT DIMENSIONALITY REDUCTION
plt.figure(figsize=(20, 15))
plt.scatter(bottleneck_representation[:,0],
bottleneck_representation[:,1],
            c = Y, s = 1, cmap = 'tab20')
plt.title('Autoencoder: 34 Hidden Layers, 10X Genomics 1.3M Mouse
Brain Cells')
plt.xlabel("Dimension 1")
plt.ylabel("Dimension 2")
plt.show()
```



مشفر تلقائي عميق لتقليل الأبعاد لخلية دماغ الفأر 10X 1.3M، تنفيذ Keras

يمكننا أن نرى أن مجموعات الخلايا يمكن تمييزها تمامًا على الرغم من أنني لم أحاول تحديدًا العثور على التكوين الأمثل للشبكة العصبية التي من المحتمل أن توفر دقة أفضل. ميزة كبيرة للمشفر التلقائي العميق هي أنه يبدو سريعًا جدًا وبالتالي قابل للقياس scalable للكميات الكبيرة من بيانات scRNAseq، فقد استغرق الأمر دقيقتين فقط على الكمبيوتر المحمول الخاص بي حتى يتقارب converge النموذج ويحصل على المخطط أعلاه.

المشفر التلقائي لـ scRNAseq مع TensorFlow

Keras رائعة وسريعة وسهلة ولكن في بعض الأحيان يكون لدي انطباع بأنني أتحكم في الشبكات العصبية بشكل أفضل باستخدام TensorFlow. على سبيل المثال، مع Keras لا يمكن للمرء أبدًا الشعور بالربط "يدويًا" manually بين العقد، وهذا هو العمق في الفهم الذي أقدر تحقيقه باستخدام TensorFlow. العيب البسيط لهذا الأخير هو أن خدعة مفيدة مثل التعلم على دفعات صغيرة -mini batch learning يجب أن يتم ترميزها يدويًا في TensorFlow ويتم تضمينها تلقائيًا في Keras على النقيض من ذلك. على أي حال، إليك كود المشفر التلقائي العميق ومخطط تقليل الأبعاد:

```
import numpy as np
import pandas as pd
import tensorflow as tf
import matplotlib.pyplot as plt
```

```
X = x_train

# DEFINE HYPERPARAMETERS
learning_rate   = 0.001
training_epochs = 500
mini_batch_size = 1024          # mini_batch_size = X.shape[0]-1
display_step    = 10            # how often to display loss and
accuracy
num_hidden_1    = 12            # 1st hidden layer num features
num_hidden_2    = 8             # 2nd hidden layer num features
num_hidden_3    = 4             # 3-d hidden layer num features
num_bottleneck  = 2             # bottleneck num features
num_input       = X.shape[1]    # scRANAseq data input (number of
genes)

# TENSORFLOW GRAPH INPUT
x = tf.placeholder("float")
y = tf.placeholder("float")

weights = {
    'encoder_h1':  tf.Variable(tf.random_normal([num_input,
num_hidden_1])),
    'encoder_h2':  tf.Variable(tf.random_normal([num_hidden_1,
num_hidden_2])),
    'encoder_h3':  tf.Variable(tf.random_normal([num_hidden_2,
num_hidden_3])),
    'bottleneck':  tf.Variable(tf.random_normal([num_hidden_3,
num_bottleneck])),
    'decoder_h1':  tf.Variable(tf.random_normal([num_bottleneck,
num_hidden_3])),
    'decoder_h2':  tf.Variable(tf.random_normal([num_hidden_3,
num_hidden_2])),
    'decoder_h3':  tf.Variable(tf.random_normal([num_hidden_2,
num_hidden_1])),
    'decoder_out': tf.Variable(tf.random_normal([num_hidden_1,
num_input])),
}
biases = {
    'encoder_b1':  tf.Variable(tf.random_normal([num_hidden_1])),
    'encoder_b2':  tf.Variable(tf.random_normal([num_hidden_2])),
    'encoder_b3':  tf.Variable(tf.random_normal([num_hidden_3])),
    'bottleneck':  tf.Variable(tf.random_normal([num_bottleneck])),
    'decoder_b1':  tf.Variable(tf.random_normal([num_hidden_3])),
    'decoder_b2':  tf.Variable(tf.random_normal([num_hidden_2])),
    'decoder_b3':  tf.Variable(tf.random_normal([num_hidden_1])),
    'decoder_out': tf.Variable(tf.random_normal([num_input])),
}

# CONSTRUCT AUTOENCODER MODEL
print("\n" + "Constructing Autoencoder Model ..." + "\n")
def encoder(x):
    layer_1 = tf.nn.elu(tf.add(tf.matmul(x, weights['encoder_h1']),
                               biases['encoder_b1']))
```

```
    layer_2 = tf.nn.elu(tf.add(tf.matmul(layer_1,
weights['encoder_h2']),
                               biases['encoder_b2']))
    layer_3 = tf.nn.elu(tf.add(tf.matmul(layer_2,
weights['encoder_h3']),
                               biases['encoder_b3']))
    bottleneck = tf.add(tf.matmul(layer_3, weights['bottleneck']),
                        biases['bottleneck'])
    return bottleneck

def decoder(x):
    layer_1 = tf.nn.elu(tf.add(tf.matmul(x, weights['decoder_h1']),
                               biases['decoder_b1']))
    layer_2 = tf.nn.elu(tf.add(tf.matmul(layer_1,
weights['decoder_h2']),
                               biases['decoder_b2']))
    layer_3 = tf.nn.elu(tf.add(tf.matmul(layer_2,
weights['decoder_h3']),
                               biases['decoder_b3']))
    layer_out = tf.nn.sigmoid(tf.add(tf.matmul(layer_3,
weights['decoder_out']),
                                     biases['decoder_out']))
    return layer_out

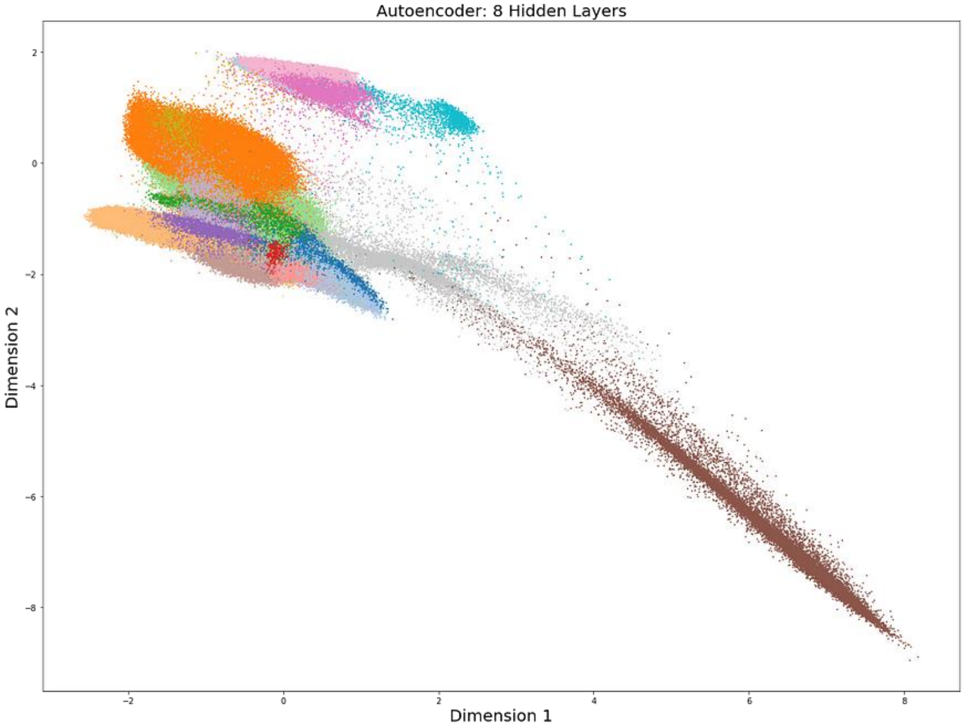
encoder_op = encoder(x)
decoder_op = decoder(encoder_op)
y_pred = decoder_op
y_true = x
cost = tf.reduce_mean(tf.pow(y_true - y_pred, 2))
optimizer = tf.train.AdamOptimizer(learning_rate).minimize(cost)

# START TRAINING AUTOENCODER
print("\n" + "Start Training Autoencoder ..." + "\n")
my_cost = []
tf.set_random_seed(12)
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    for epoch in range(training_epochs):
        pos = 0
        idx = np.arange(X.shape[0])
        my_cost_mini_batch = []
        for _ in range(10000000):
            #print('Mini-batch {0} - {1}'.format(pos,pos +
mini_batch_size))
            if pos + mini_batch_size >= X.shape[0]:
                break
            batch_x = X[idx[range(pos, pos + mini_batch_size)],:]
            c,_ = sess.run([cost, optimizer], feed_dict={x:
batch_x})
            my_cost_mini_batch.append(c)
            pos = pos + mini_batch_size
        my_cost.append(np.mean(my_cost_mini_batch))
        if (epoch + 1) % display_step == 0:
            print("Epoch:", '%04d' % (epoch + 1),
```

```
                   "cost = ",
"{:.9f}".format(np.mean(my_cost_mini_batch)))
    pred = sess.run(encoder(x), feed_dict={x: X})

# PLOT LOSS FUNCTION
plt.figure(figsize=(20, 15))
plt.plot(range(training_epochs), my_cost)
plt.xlabel("Epoch",fontsize = 20)
plt.ylabel("Loss",fontsize = 20, rotation = 1)
plt.show()

# VISUALIZE AUTOENCODER BOTTLENECK
plt.figure(figsize=(20, 15))
plt.scatter(pred[:,0], pred[:,1], c = Y, s = 1, cmap = 'tab20')
plt.title('Autoencoder: 8 Hidden Layers', fontsize = 20)
plt.xlabel("Dimension 1", fontsize = 20)
plt.ylabel("Dimension 2", fontsize = 20)
plt.show()
```



مشفر تلقائي عميق لتقليل الأبعاد لخلايا دماغ الفأر 1.3M 10X، تنفيذ TensorFlow

مرة أخرى، يبدو التمثيل منخفض الأبعاد واعداً. مع بعض التغيير والتبديل في التكوين والمعلمات الفائقة الأخرى، يمكن للمرء الحصول على دقة أفضل بكثير. مرة أخرى، بالنسبة إلى Keras، فإن تطبيق المشفر التلقائي العميق هذا سريع للغاية، فقط بضع دقائق مقارنة بالساعات المستخدمة بواسطة FItSNE و

UMAP و SCVIS لإنتاج هذا النوع من مخطط تقليل الأبعاد. إذا كانت هناك حاجة إلى إجراء إعادة التشكيل resampling لسبب ما لتحليلك، فلن يكون ذلك ممكنًا للغاية مع FItSNE و UMAP و SCVIS ولكن يجب أن يكون من السهل جدًا تنفيذه باستخدام التعلم العميق.

الاستنتاج

لقد تعلمنا هنا أن تسلسل الحمض النووي الريبي أحادي الخلية (scRNAseq) يحسن بسرعة فهمنا للتنوع الوظيفي بين الخلايا البيولوجية. ربما يكون تقليل الأبعاد هو أهم أداة تحليلية وراء تحليل scRNAseq النموذجي. تواجه تقنية تقليل الأبعاد القياسية الذهبية tSNE حاليًا صعوبات في قابلية التوسع بسبب الكميات الكبيرة من البيانات التي تقدمها تقنيات scRNAseq. يوفر التعلم العميق عبر Autoencoder و UMAP حاليًا أكثر الطرق مرونة وقابلية للتطوير للحصول على تمثيلات منخفضة الأبعاد لبيانات scRNAseq ومن المرجح أن يحل محل tSNE في المستقبل. أخيرًا، تعلمنا كيفية تنفيذ المشفر التلقائي العميق لمجموعة بيانات scRNAseq لخلايا دماغ الفأر العملاقة 10X Genomics 1.3M باستخدام Keras و TensorFlow.

المصدر:

https://towardsdatascience.com/deep-learning-for-single-cell-biology-935d45064438

10) التعلم العميق لتكامل البيانات Deep Learning for Data Integration

(التأثيرات التآزرية لتكامل البيانات مع التعلم العميق)

هذه هي المقالة الثالثةفي سلسلة التعلم العميق لعلوم الحياة.في المنشورين السابقين، أوضحت كيفية استخدام التعلم العميق على الحمض النووي القديم والتعلم العميق لبيولوجيا الخلية المفردة. سنناقش الآن كيفية استخدام مصادر متعددة للمعلومات البيولوجية، وبيانات OMICs، من أجل تحقيق نمذجة أكثر دقة للأنظمة البيولوجية عن طريق التعلم العميق.

استفادت الأبحاث البيولوجية والطبية الحيوية بشكل كبيرفي العقد الماضي من التقدم التكنولوجي الذي يقدم تسلسل الحمض النووي (الجينوميات genomics)، والتعبير الجيني gene expression (النسخ transcriptomics)، ووفرة البروتين protein abundance (البروتينات proteomics) والعديد من المستويات الأخرى من المعلومات البيولوجية التي يشار إليها عادة باسم OMICs. على الرغم من أن طبقات OMIC الفردية قادرة على الإجابة على العديد من الأسئلة البيولوجية المهمة، إلا أن دمجها وتأثيراتها التآزرية synergistic effects الناتجة عن تكاملها تعد برؤى جديدةفي سلوك الأنظمة البيولوجية مثل الخلايا والأنسجة والكائنات الحية. لذلك يمثل تكامل OMICs (OMICs integration) التحدي المعاصرفي علم الأحياء Biology والطب الحيوي Biomedicine.

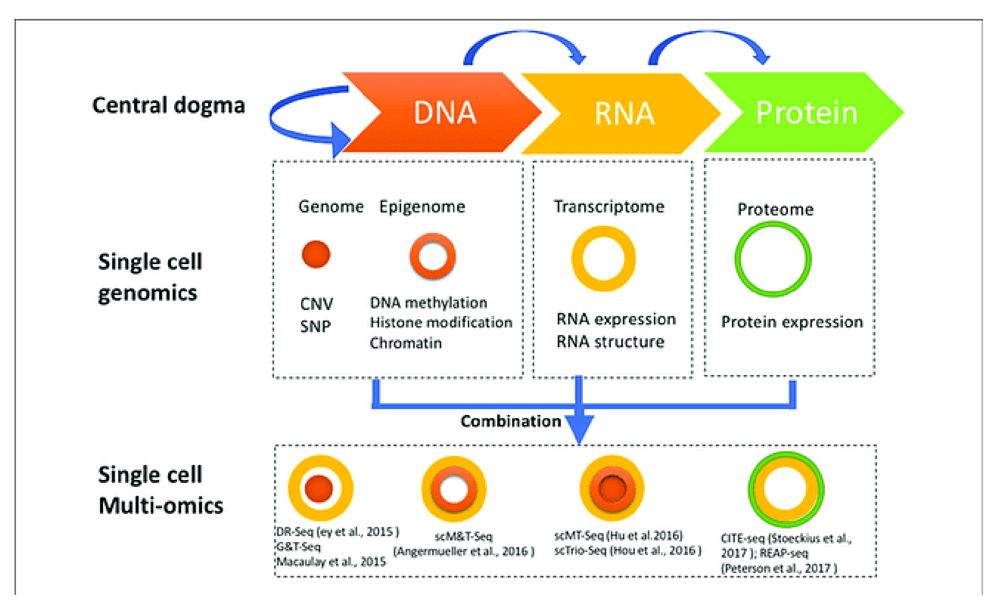
في هذه المقالة، سأستخدم التعلم العميق Deep Learning مع Keras وأظهر كيف أن تكامل بيانات OMICs المتعددة (multi-OMICs data) يكشف عن أنماط مخفية غير مرئيةفي OMICs الفردية.

الخلايا المفردة تصنع البيانات الضخمة

مشكلة تكامل البيانات data integration ليست جديدة تمامًا لعلوم البيانات. تخيل أننا نعلم أن شخصًا ما ينظر إلى صور معينة ويقرأ نصوصًا معينة ويستمع إلى موسيقى معينة. تعد الصورة والنص والصوت أنواعًا مختلفة جداً من البيانات، ومع ذلك يمكننا محاولة دمج combine هذه الأنواع من البيانات من أجل البناء على سبيل المثال نظام توصية recommender system أفضل يحقق دقة أعلىفي التقاط اهتمامات الشخص. بالنسبة إلى علم الأحياء والطب الحيوي، لم تصل فكرة تكامل البيانات إلى هنا إلا مؤخرًا، ومع ذلك فقد تم تطويرها بنشاط مع الزاوية البيولوجية مما أدى إلى العديد من المنهجيات المثيرة للاهتمام مثل mixOmics و MOFA وانصهار شبكة التشابه Similarity Network Fusion (SNF) و DISCO / JIVE / OnPLS، شبكات بايزي Bayesian Networks إلخ.



إحدى المشكلات التي تواجهها جميع أساليب OMIC التكاملية المذكورة أعلاه هي لعنة الأبعاد curse of dimensionality، أي عدم القدرة على العمل في مساحة عالية الأبعاد مع عدد محدود من الملاحظات الإحصائية statistical observations، وهو إعداد نموذجي لتحليل البيانات البيولوجية. هذا هو المكان الذي تكون فيه تقنيات الخلايا أحادية الخلية مفيدة للغاية لأنها تقدم مئات الآلاف وحتى الملايين من الملاحظات الإحصائية (الخلايا) كما ناقشناهافي المقالة السابقة، وبالتالي توفر بيانات كبيرة حقًا مثالية للتكامل Big Data ideal for integration.



تقنيات خلية واحدة متعددة OMICs.

إنه لأمر مثير للغاية أن تقنيات الخلية المفردة متعددة OMIC مثل CITEseq و scNMTseq توفر مستويين وثلاثة مستويات من المعلومات البيولوجية biological information، على التوالي، من نفس الخلايا بالضبط.

تكامل بيانات CITEseq مع التعلم العميق

سنقوم هنا بإجراء تكامل غير خاضع للإشراف unsupervised integration لبيانات النسخ transcriptomics للخلايا المفردة (scRNAseq) والبروتيوميات (scProteomics) من CITEseq، 8 617 خلية أحادية النواة لدم الحبل السري cord blood mononuclear cells (CBMC) ، باستخدام المشفر التلقائي Autoencoder التي تناسب بشكل مثالي التقاط الطبيعة غير الخطية للغاية لبيانات OMICs أحادية الخلية . لقد قمنا بتغطية مزايا استخدام المشفر التلقائي لبيولوجيا الخلية المفردةفي المنشور السابق، ولكن باختصار ترتبط بحقيقة أن تحليل الخلية المفردة غير خاضع للإشراف بشكل أساسي. نبدأ بتنزيل بيانات CITEseq من هنا، وقراءتها مع Pandas وتحويل السجل log-transforming، وهو ما يعادل تطبيعًا معتدل mild normalization. كالعادة، الصفوف عبارة عن خلايا، والأعمدة هي mRNA أو ميزات بروتينية protein features، والعمود الأخير يتوافق مع شرح الخلية cell annotation.

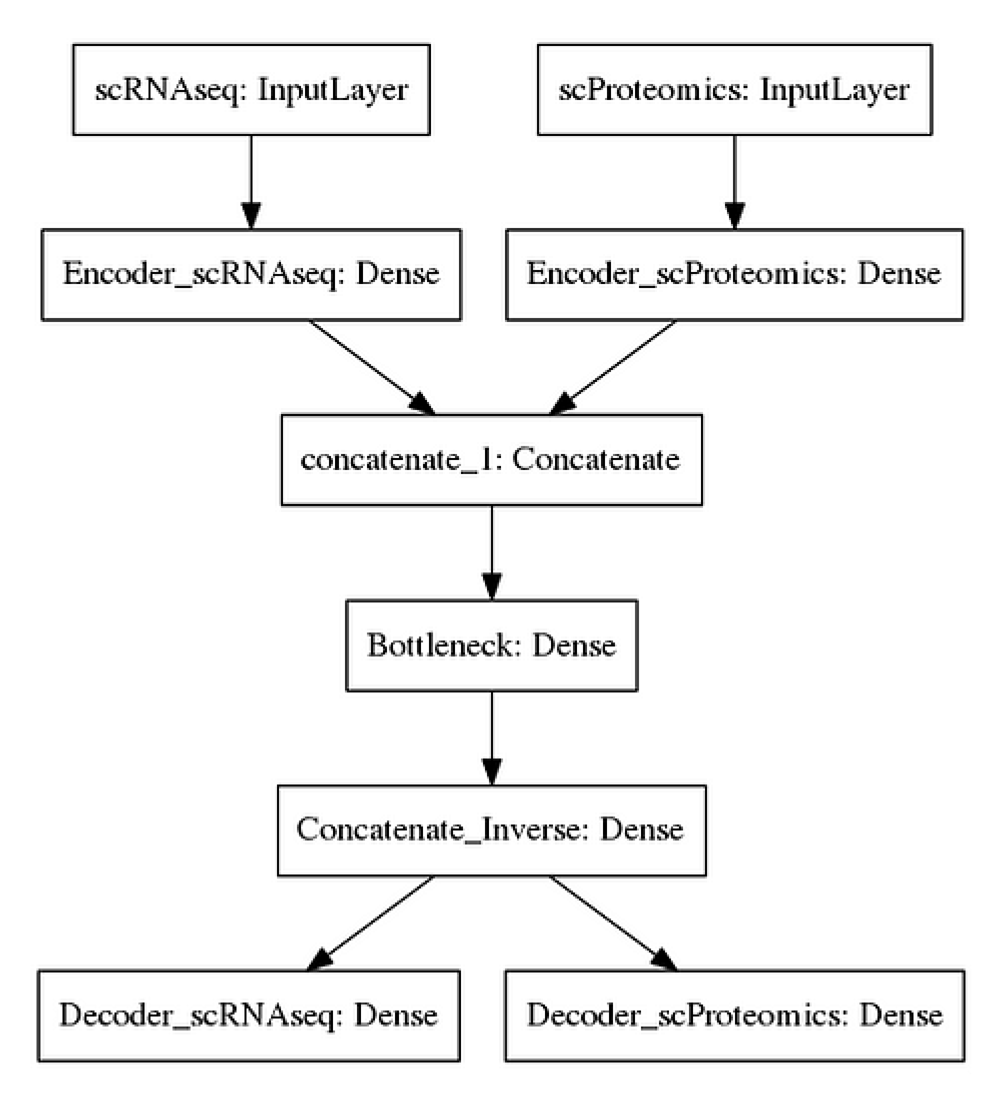
```
import numpy as np
import pandas as pd
from keras.models import Model
import matplotlib.pyplot as plt
from sklearn.manifold import TSNE
from keras.utils import plot_model
from keras.layers import Input, Dense
from keras.layers.merge import concatenate

scRNAseq = pd.read_csv('scRNAseq.txt',sep='\t')
scProteomics = pd.read_csv('scProteomics.txt',sep='\t')

X_scRNAseq = scRNAseq.values[:,0:(scRNAseq.shape[1]-1)]
Y_scRNAseq = scRNAseq.values[:,scRNAseq.shape[1]-1]
X_scProteomics = scProteomics.values[:,0:(scProteomics.shape[1]-1)]
Y_scProteomics = scProteomics.values[:,scProteomics.shape[1]-1]

X_scRNAseq = np.log(X_scRNAseq + 1)
X_scProteomics = np.log(X_scProteomics + 1)
```

سنقوم الآن ببناء نموذج المشفر التلقائي مع 4 طبقات مخفية باستخدام Keras الوظيفية API. يحتوي Autoencoder على مدخلين، واحد لكل طبقة من المعلومات، أي scRNAseq وscProteomics، ومخرجات مقابلة تهدف إلى إعادة بناء المدخلات. يتم تحويل طبقتي الإدخال input layers بشكل خطي بشكل منفصل في الطبقة المخفية الأولى first hidden layer (ما يعادل تقليل أبعاد PCA) قبل أن يتم تجميعهمافي الطبقة المخفية الثانية second hidden layer. أخيرًا، تتم معالجة OMICs المدمجة من خلال عنق الزجاجة bottleneckفي المشفر التلقائي، وأخيرًا يتم إعادة بناء الأبعاد تدريجيًا إلى الأبعاد الأولية وفقًا لتماثل "الفراشة butterfly " النموذجي للمشفرات التلقائية.



التكامل غير الخاضع للإشراف لبيانات CITEseq

في الكود الخاص بـ Autoencoder أدناه، من المهم ملاحظة أن الطبقة المخفية الأولى تفرض تقليلًا شديدًافي الأبعاد على scRNAseq من 977 إلى 50 جينًا، بينما تترك scProteomics كما هي تقريبًا، أي تقليل الأبعاد من 11 إلى 10. يقلل إجمالي الأبعاد الستين بعد التسلسل وصولاً إلى 50 متغيرًا كامنًا latent variables والتي تمثل مجموعات من ميزات كل من mRNA والبروتين.

```
# Input Layer
ncol_scRNAseq = X_scRNAseq.shape[1]
input_dim_scRNAseq = Input(shape = (ncol_scRNAseq, ), name =
"scRNAseq")
ncol_scProteomics = X_scProteomics.shape[1]
```

```
input_dim_scProteomics = Input(shape = (ncol_scProteomics, ), name
= "scProteomics")

# Dimensions of Encoder for each OMIC
encoding_dim_scRNAseq = 50
encoding_dim_scProteomics = 10

# Encoder layer for each OMIC
encoded_scRNAseq = Dense(encoding_dim_scRNAseq, activation =
'linear',
                         name =
"Encoder_scRNAseq")(input_dim_scRNAseq)
encoded_scProteomics = Dense(encoding_dim_scProteomics, activation
= 'linear',
                             name =
"Encoder_scProteomics")(input_dim_scProteomics)

# Merging Encoder layers from different OMICs
merge = concatenate([encoded_scRNAseq, encoded_scProteomics])

# Bottleneck compression
bottleneck = Dense(50, kernel_initializer = 'uniform', activation =
'linear',
                   name = "Bottleneck")(merge)

#Inverse merging
merge_inverse = Dense(encoding_dim_scRNAseq +
encoding_dim_scProteomics,
                      activation = 'elu', name =
"Concatenate_Inverse")(bottleneck)

# Decoder layer for each OMIC
decoded_scRNAseq = Dense(ncol_scRNAseq, activation = 'sigmoid',
                         name = "Decoder_scRNAseq")(merge_inverse)
decoded_scProteomics = Dense(ncol_scProteomics, activation =
'sigmoid',
                             name =
"Decoder_scProteomics")(merge_inverse)

# Combining Encoder and Decoder into an Autoencoder model
autoencoder = Model(input = [input_dim_scRNAseq,
input_dim_scProteomics],
                    output = [decoded_scRNAseq,
decoded_scProteomics])

# Compile Autoencoder
autoencoder.compile(optimizer = 'adam',
                    loss={'Decoder_scRNAseq': 'mean_squared_error',
                          'Decoder_scProteomics':
'mean_squared_error'})
autoencoder.summary()
```

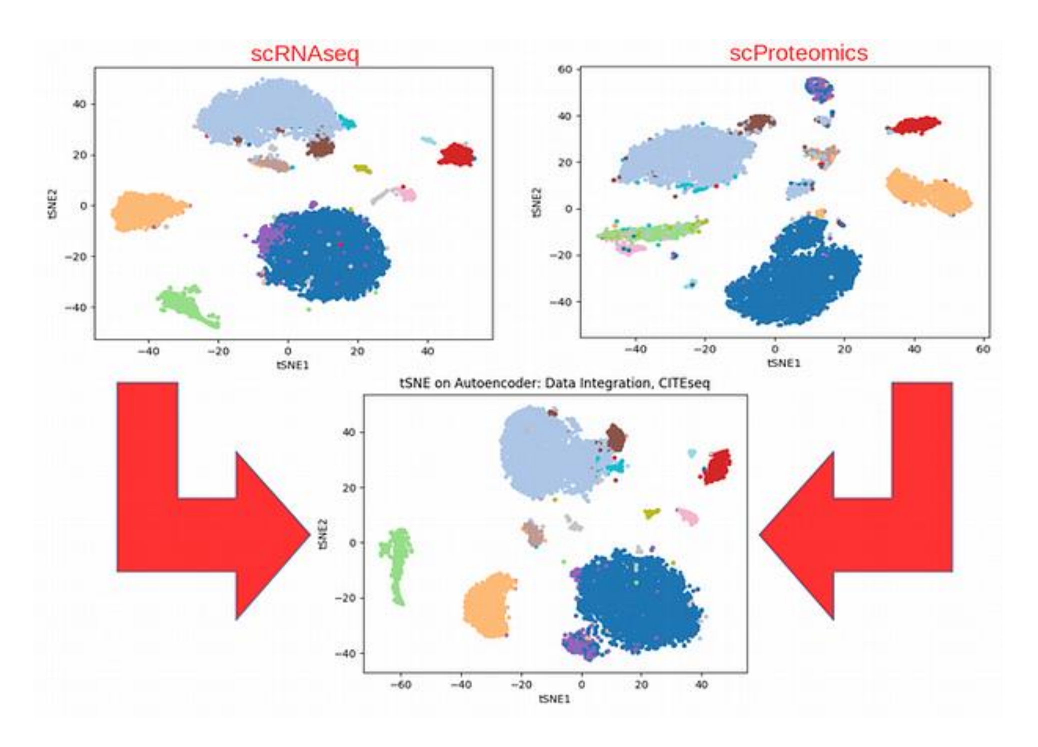
شيء مفيد للغاية هنا هو أنه يمكننا تعيين دوال الخطأ loss functions المختلفة إلى OMICs القادمة من توزيعات إحصائية مختلفة، على سبيل المثال من خلال الجمع بين البيانات الفئوية categorical

والمستمرة continuous، يمكننا تطبيق الإنتروبيا المتقاطعة الفئوية categorical cross entropy ومتوسط الخطأ التربيعي mean squared error، على التوالي. شيء رائع آخر حول تكامل البيانات عبر Autoencoders هو أن جميع OMICs تعرف بعضها البعض حيث يتم تحديث أوزان كل عقدة node / ميزة feature من خلال الانتشار الخلفي back propagation في سياق بعضها البعض.
أخيرًا، دعونا ندرب Autoencoder وإدخال عنق الزجاجة في tSNE من أجل التصور:

```
# Autoencoder training
estimator = autoencoder.fit([X_scRNAseq, X_scProteomics],
                            [X_scRNAseq, X_scProteomics],
                            epochs = 100, batch_size = 128,
                            validation_split = 0.2, shuffle = True,
verbose = 1)
print("Training Loss: ",estimator.history['loss'][-1])
print("Validation Loss: ",estimator.history['val_loss'][-1])
plt.plot(estimator.history['loss'])
plt.plot(estimator.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train','Validation'], loc = 'upper right')
plt.show()

# Encoder model
encoder = Model(input = [input_dim_scRNAseq,
input_dim_scProteomics],
                output = bottleneck)
bottleneck_representation = encoder.predict([X_scRNAseq,
X_scProteomics])

# tSNE on Autoencoder bottleneck representation
model_tsne_auto = TSNE(learning_rate = 200, n_components = 2,
random_state = 123,
                       perplexity = 90, n_iter = 1000, verbose = 1)
tsne_auto =
model_tsne_auto.fit_transform(bottleneck_representation)
plt.scatter(tsne_auto[:, 0], tsne_auto[:, 1], c = Y_scRNAseq, cmap
= 'tab20', s = 10)
plt.title('tSNE on Autoencoder: Data Integration, CITEseq')
plt.xlabel("tSNE1")
plt.ylabel("tSNE2")
plt.show()
```

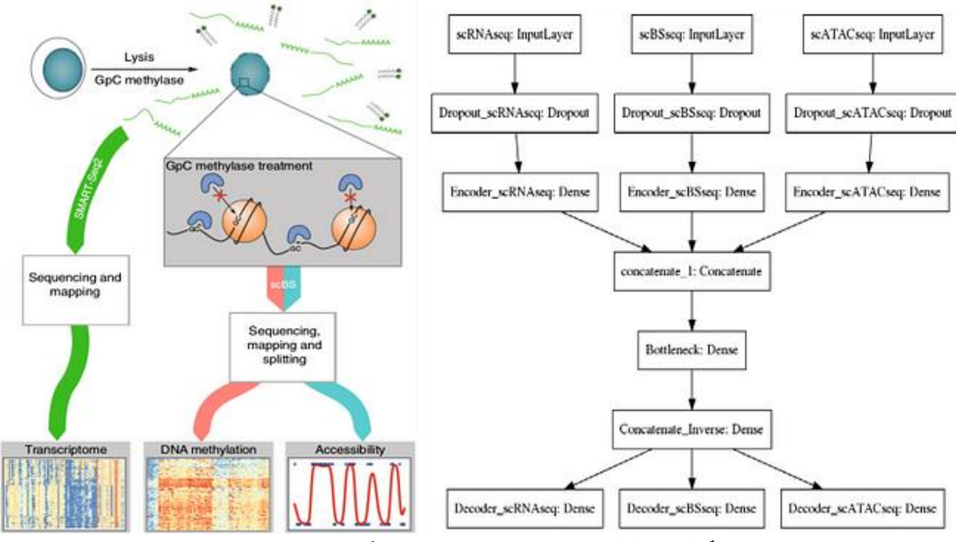


تأثير تكامل بيانات CITEseq: لرؤية الأنماط غير المرئيةفي OMICs الفردية

بمقارنة مخططات tSNE التي تم الحصول عليها باستخدام OMICs الفردية مع tSNE على عنق الزجاجةفي المشفر التلقائي الذي يجمع البيانات، يمكننا أن نرى على الفور أن التكامل متوسط إلى حد ما ويعزز OMICs الفردية. على سبيل المثال، سيكون من الصعب اكتشاف الكتلة الأرجواني باستخدام بيانات scRNAseq وحدها لأنها لا تختلف عن مجموعة الخلايا الزرقاء، ولكن بعد التكامل يمكن تمييز مجموعة الخلايا الأرجواني بسهولة. هذه هي قوة تكامل البيانات!

تكامل بيانات scNMTseq مع التعلم العميق

بينما يشتمل CITEseq على مستويين من الخلايا المفردة من المعلومات (transcriptomics و Protomics)، فإن تقنية رائعة أخرى، scNMTseq، تقدم ثلاثة OMICs من نفس الخلايا البيولوجية: 1) transcriptomics (scRNAseq)، 2) methylation pattern (scBSseq) ، و 3) open chromatin regions (scATACseq). يمكن تنزيل البيانات الأولية من هنا.



تكامل بيانات scNMTseq مع Autoencoder

إن بُنية Autoencoder مماثلة لتلك المستخدمةفي CITEseq مع خصوصية واحدة فقط: يتم استخدام تسوية التسرب Dropout regularization على طبقات الإدخال input layers. هذا يرجع إلى حقيقة أن لدينا 120 خلية فقط متسلسلة بينما أبعاد مساحة الميزة هي عشرات الآلاف، لذلك نحن بحاجة إلى تطبيق التنظيم regularization للتغلب على لعنة الأبعاد. لاحظ أن هذا لم يكن ضروريًا لـ CITEseq حيث كان لدينا حوالي K8 خلية و ~ K1 ميزات، لذلك الوضع المعاكس تمامًا. ومع ذلك، فإن scNMTseq بشكل عام ليست حالة سهلة لتكامل البيانات، وأعتقد اعتقادًا راسخًا أن هذه مجرد بداية لعصر خلية مفردة متعددة الـ OMICs وأن العديد من الخلايا ستصل قريبًا من هذه التقنية المثيرة، لذلك من الأفضل أن تكون مستعدًا.

```
import numpy as np
import pandas as pd
from umap import UMAP
from keras.models import Model
import matplotlib.pyplot as plt
from keras.layers.merge import concatenate
from keras.layers import Input, Dense, Dropout

################# READ AND TRANSFORM DATA #################
scRNAseq = pd.read_csv('scRNAseq.txt',sep='\t')
scBSseq = pd.read_csv('scBSseq.txt',sep='\t')
scATACseq = pd.read_csv('scATACseq.txt',sep='\t')

X_scRNAseq = scRNAseq.values[:,0:(scRNAseq.shape[1]-1)]
Y_scRNAseq = scRNAseq.values[:,scRNAseq.shape[1]-1]
X_scBSseq = scBSseq.values[:,0:(scBSseq.shape[1]-1)]
Y_scBSseq = scBSseq.values[:,scBSseq.shape[1]-1]
X_scATACseq = scATACseq.values[:,0:(scATACseq.shape[1]-1)]
```

```
Y_scATACseq = scATACseq.values[:,scATACseq.shape[1]-1]

X_scRNAseq = np.log(X_scRNAseq + 1)
X_scBSseq = np.log(X_scBSseq + 1)
X_scATACseq = np.log(X_scATACseq + 1)

######################## AUTOENCODER ########################
# Input Layer
ncol_scRNAseq = X_scRNAseq.shape[1]
input_dim_scRNAseq = Input(shape = (ncol_scRNAseq, ), name =
"scRNAseq")
ncol_scBSseq = X_scBSseq.shape[1]
input_dim_scBSseq = Input(shape = (ncol_scBSseq, ), name =
"scBSseq")
ncol_scATACseq = X_scATACseq.shape[1]
input_dim_scATACseq = Input(shape = (ncol_scATACseq, ), name =
"scATACseq")

encoding_dim_scRNAseq = 30
encoding_dim_scBSseq = 30
encoding_dim_scATACseq = 30

# Dropout on Input Layer
dropout_scRNAseq = Dropout(0.2, name =
"Dropout_scRNAseq")(input_dim_scRNAseq)
dropout_scBSseq = Dropout(0.2, name =
"Dropout_scBSseq")(input_dim_scBSseq)
dropout_scATACseq = Dropout(0.2, name =
"Dropout_scATACseq")(input_dim_scATACseq)

# Encoder layer for each OMIC
encoded_scRNAseq = Dense(encoding_dim_scRNAseq, activation = 'elu',
                         name =
"Encoder_scRNAseq")(dropout_scRNAseq)
encoded_scBSseq = Dense(encoding_dim_scBSseq, activation = 'elu',
                        name = "Encoder_scBSseq")(dropout_scBSseq)
encoded_scATACseq = Dense(encoding_dim_scATACseq, activation =
'elu',
                          name =
"Encoder_scATACseq")(dropout_scATACseq)

# Merging Encoder layers from different OMICs
merge = concatenate([encoded_scRNAseq, encoded_scBSseq,
encoded_scATACseq])

# Bottleneck compression
bottleneck = Dense(50, kernel_initializer = 'uniform', activation =
'linear',
                   name = "Bottleneck")(merge)

#Inverse merging
merge_inverse = Dense(encoding_dim_scRNAseq + encoding_dim_scBSseq
+
                      encoding_dim_scATACseq,
```

```
                        activation = 'elu', name =
"Concatenate_Inverse")(bottleneck)

# Decoder layer for each OMIC
decoded_scRNAseq = Dense(ncol_scRNAseq, activation = 'sigmoid',
                           name = "Decoder_scRNAseq")(merge_inverse)
decoded_scBSseq = Dense(ncol_scBSseq, activation = 'sigmoid',
                          name = "Decoder_scBSseq")(merge_inverse)
decoded_scATACseq = Dense(ncol_scATACseq, activation = 'sigmoid',
                            name =
"Decoder_scATACseq")(merge_inverse)

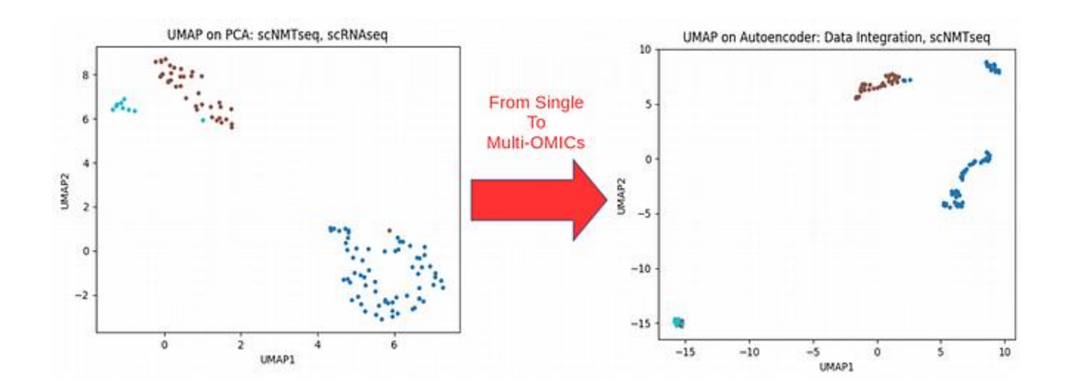
# Combining Encoder and Decoder into an Autoencoder model
autoencoder = Model(input = [input_dim_scRNAseq, input_dim_scBSseq,
                             input_dim_scATACseq],
                    output = [decoded_scRNAseq, decoded_scBSseq,
decoded_scATACseq])

# Compile Autoencoder
autoencoder.compile(optimizer = 'adam',
                    loss={'Decoder_scRNAseq': 'mean_squared_error',
                          'Decoder_scBSseq': 'binary_crossentropy',
                          'Decoder_scATACseq':
'binary_crossentropy'})
autoencoder.summary()

# Autoencoder training
estimator = autoencoder.fit([X_scRNAseq, X_scBSseq, X_scATACseq],
                            [X_scRNAseq, X_scBSseq, X_scATACseq],
epochs = 130,
                            batch_size = 16, validation_split =
0.2,
                            shuffle = True, verbose = 1)
print("Training Loss: ",estimator.history['loss'][-1])
print("Validation Loss: ",estimator.history['val_loss'][-1])
plt.plot(estimator.history['loss']);
plt.plot(estimator.history['val_loss'])
plt.title('Model Loss'); plt.ylabel('Loss'); plt.xlabel('Epoch')
plt.legend(['Train','Validation'], loc = 'upper right')

# Encoder model
encoder = Model(input = [input_dim_scRNAseq, input_dim_scBSseq,
                         input_dim_scATACseq], output = bottleneck)
bottleneck_representation = encoder.predict([X_scRNAseq, X_scBSseq,
X_scATACseq])

############### UNIFORM MANIFOLD APPROXIMATION AND PROJECTION
(UMAP) ###############
model_umap = UMAP(n_neighbors = 11, min_dist = 0.1, n_components =
2)
umap = model_umap.fit_transform(bottleneck_representation)
plt.scatter(umap[:, 0], umap[:, 1], c = Y_scRNAseq, cmap = 'tab10',
s = 10)
plt.title('UMAP on Autoencoder: Data Integration, scNMTseq')
plt.xlabel("UMAP1"); plt.ylabel("UMAP2")
```



الجمع بين النسخ transcriptomics مع معلومات التخلق epigenetics information لـ scNMTseq

هنا بدافع الفضول، قمت بتغذية عنق الزجاجة لـ Autoencoder الذي يجمع ثلاثة scNMTseq OMICsفي تقنية التقريب المتشعب الموحد والإسقاط Uniform Manifold Approximation and Projection (UMAP) غير الخطي لتقليل الأبعاد التي يبدو أنها تتفوق على tSNE من حيث قابلية التوسع scalability لكميات كبيرة من البيانات. يمكننا أن نرى على الفور أن المجموعة الزرقاء المتجانسة من حيث التعبير الجيني تنقسم إلى مجموعتين عندما يتم دمج scRNAseq مع معلومات التخلق epigenetics information من نفس الخلايا (scBSseq و scATACseq). لذلك يبدو أننا قد التقطنا عدم تجانس heterogeneity جديد بين الخلايا والذي كان مخفيًا عند النظر فقطفي بيانات scRNAseq للتعبير الجيني. هل يمكن أن تكون هذه طريقة جديدة لتصنيف الخلايا عبر المجموعات السكانية باستخدام التعقيد الكامل لبيولوجيتها؟ إذاكان الأمر كذلك، فإن السؤال يأتي: ما هو عدد الخلايا أو نوع الخلية؟ لا أعرف إجابة هذا السؤال.

الاستنتاج

لقد تعلمنا هنا أن المصادر المتعددة للمعلومات الجزيئية والسريرية أصبحت شائعةفي علم الأحياء والطب الحيوي بفضل التقدم التكنولوجي الحديث. لذلك فإن تكامل البيانات هو الخطوة المنطقية التالية التي توفر فهمًا أكثر شمولاً للعمليات البيولوجية من خلال الاستفادة من التعقيد الكامل للبيانات. يعتبر إطار عمل التعلم العميق مناسبًا بشكل مثالي لتكامل البيانات نظرًا لتحديثه "التكاملي integrative" الحقيقي للمعلمات من خلال الانتشار الخلفي back propagation عندما تتعلم أنواع البيانات المتعددة المعلومات من بعضها البعض. لقد أوضحت أن تكامل البيانات يمكن أن يؤدي إلى اكتشافات لأنماط جديدةفي البيانات لم تكن موجودة من قبلفي أنواع البيانات الفردية.

المصدر:

https://medium.com/towards-data-science/deep-learning-for-data-integration-46d51601f781

11) التعلم العميق للتشخيصات السريرية Deep Learning for Clinical Diagnostics

(إجراء تنبؤات طبية حيوية أكثر أمانًا باستخدام التعلم العميق)

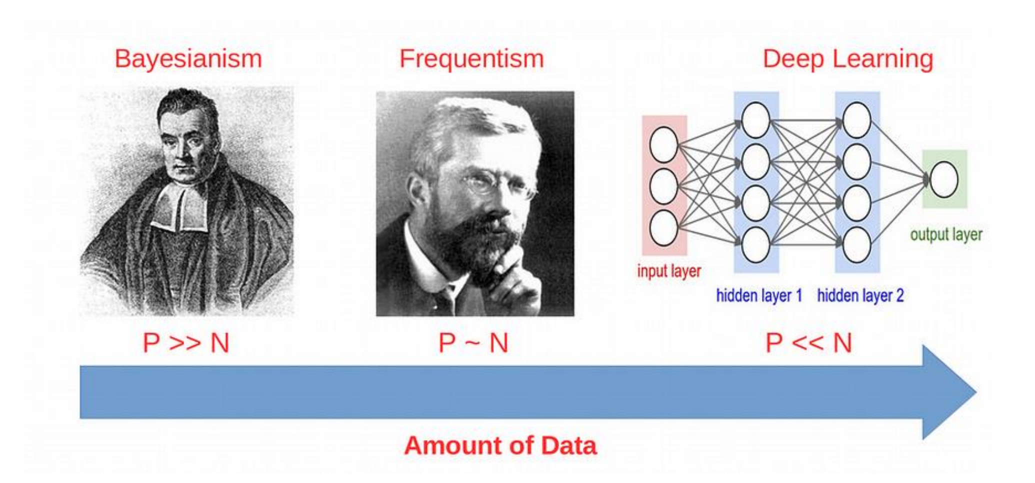
هذه هي المقالة الرابعةفي سلسلة التعلم العميق لعلوم الحياة.في المنشورات السابقة، أوضحت كيفية استخدام التعلم العميق على الحمض النووي القديم والتعلم العميق لبيولوجيا الخلية المفردة والتعلم العميق لتكامل البيانات. نحن الآن بصدد الغوص في الطب الحيوي Biomedicine ومعرفة لماذا وكيف يجب أن نستخدم التعلم العميق البايزي Bayesian Deep Learning من أجل سلامة المرضى.

قدم تسلسل الجيل التالي Next Generation Sequencing (NGS) تقدمًا كبيرًا لفهمنا للآليات المسببة للأمراض التي تؤدي إلى أمراض بشرية شائعة. ومع ذلك، لا تزال كمية البيانات تشكل عنق الزجاجة للتحليل في الطب الحيوي. على عكس علم البيانات Data Science، فإن الملايين من الأمثلة غير شائعة إلى حد مافي الطب الحيوي بينما البيانات عالية الأبعاد نموذجية تمامًا، لذلك فإن التعلم الآلي له تطبيقات محدودة للغايةفي الطب الحيوي. يؤدي نقص البيانات ومساحة المعلمات عالية الأبعاد إلى إعاقة الدقةفي التشخيص السريري clinical diagnostics مما يؤدي إلى الكثير من التنبؤات الخاطئة التي لا تثبتفي التجارب السريرية clinical trials. عندما تكون البيانات متناثرة sparse/ نادرة scarce/ صاخبة noisy وعالية الأبعاد high-dimensional، يساعد الإحصاء البايزي Bayesian Statistics في عمل تنبؤات قابلة للتعميم generalizable predictions.

سنناقش هنا كيفية تنفيذ التعلم العميق البايزي Bayesian Deep Learning مع PyMC3 من أجل ضمان سلامة المريض وتقديم تنبؤات أكثر دقة وذكاء للتشخيص السريري.

لماذا تكون بايزي عند تشغيل التعلم العميق؟

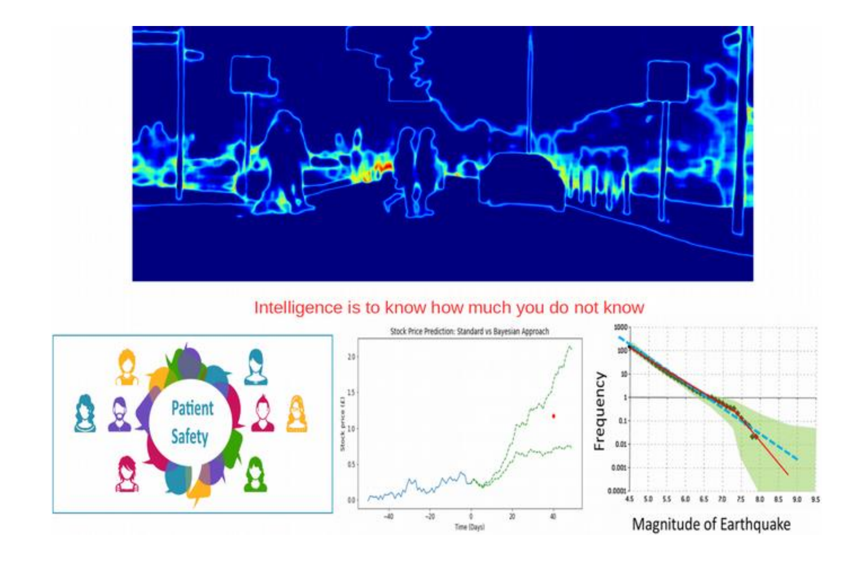
في المنشور السابق أوضحت أنه عند إجراء تحليل إحصائي statistical analysis، يجب أن تولي اهتمامًا خاصًا للتوازن بين عدد الملاحظات الإحصائية statistical observations، N، وأبعاد المساحة الخاصة بك، أي عدد الميزات number of features، P. اعتمادًا على كمية البيانات، أنت يمكن الاختيار بين الإحصاء البايزي Bayesian Statistics والإحصاء المتكرر Frequentist Statistics والتعلم الآلي / العميق Machine/Deep Learning.



من المنطقي استخدام التعلم العميق فقط عندما يكون لديك الكثير من البيانات

لذلك من المنطقي استخدام التعلم العميق عندما يكون لديك الكثير من البيانات لأنه يمكنك التخلي عن عالم الجبر الخطي الممل والقفز إلى حفرة الأرانبفي الرياضيات غير الخطية.في المقابل، يعمل الطب الحيوي عادةفي الحد المقابل، P >> N، ويحتاج إلى احتمالات سابقة Priors للتعويض عن نقص البيانات. هذا هو السبب الأول الذي يجعل التحليل الطبي الحيوي Biomedical analysis يجب أن يكون بايزي.

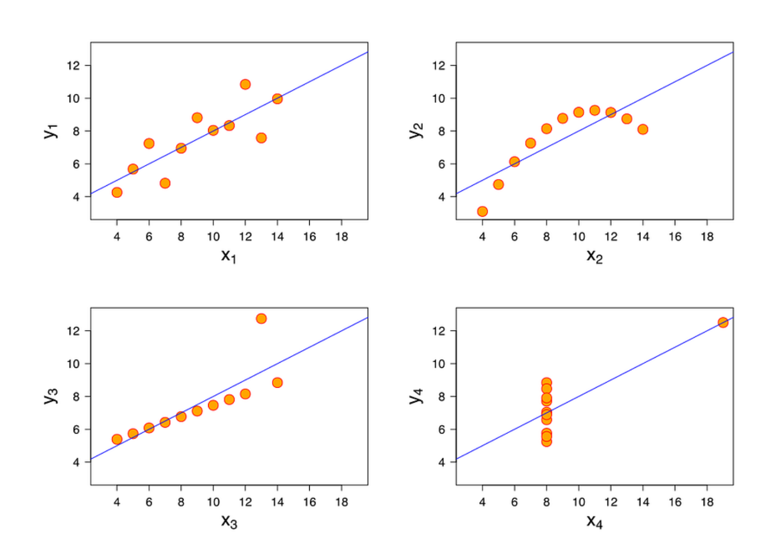
تخيل الآن للحظة أنك حصلت على بعض البيانات الطبية الحيوية الضخمة، فهذا أمر غير شائع ولكنه ليس مستحيلًا إذا عمل المرء مع التصوير أو بيولوجيا الخلية المفردة، هنا يمكنك ويجب عليك القيام بالتعلم العميق. لكن لماذا تريد أن تكون بايزيفي هذه الحالة؟



هنا يأتي السبب الثاني: ضرورة إنشاء تنبؤات أقل فئوية less categorical (مقارنة بالتعلم العميق القائم على التكرار التقليدي traditional Frequentist) من خلال دمج أوجه عدم اليقين uncertainties في النموذج. هذا له أهمية كبيرة بالنسبة للمناطق ذات الأسعار الباهظة للتنبؤات الخاطئة مثل السيارات ذاتية القيادة self-driving cars، ونمذجة سوق الأسهم modelling stock market، والزلازل earthquakes، وخاصة التشخيصات السريرية clinical diagnostics.

لماذا لا يتم التحليل التكراري للطب الحيوي؟

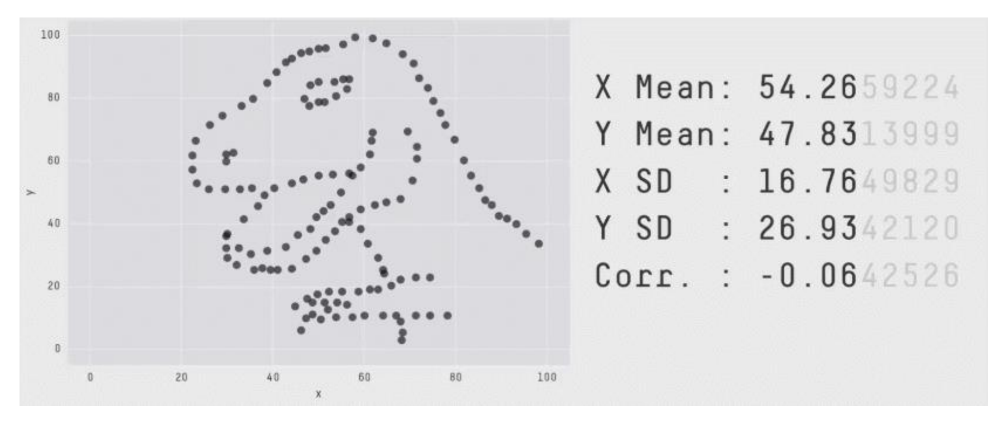
هناك العديد من الأسباب التي يجب أن تكون حذرًا عند تطبيق الإحصائيات التكرارية Frequentist Statistics على التشخيص السريري. يعتمد بشكل كبير على افتراض الحالة الطبيعية وبالتالي فهو حساس للقيم المتطرفة outliers، ويعمل مع الإحصاءات الوصفية descriptive statistics التي لا تعكس دائمًا توزيعات البيانات الأساسية وبالتالي تفشل في التقاط الفرق بين مجموعات البيانات في مجموعة أنسكومب الرباعية Anscombe's quartet بشكل صحيح.



في المقابل، ستؤدي النمذجة الاحتمالية البايزية Bayesian probabilistic modelling لمجموعات بيانات أنسكومب إلى اختلافات كبيرة في توزيعات الاحتمالات.

الذكاء هو أن تعرف مقدار ما لا تعرفه

هناك بعض الأمثلة الشهيرة التي يشار إليها عادةً باسم Data Saurus والتي توضح أيضًا أن الإحصاء التكراري لا يمكنه التقاط الفرق بين مجموعات العينات ذات الإحصائيات الوصفية المتطابقة مثل المتوسط mean أو الانحراف المعياري standard deviation أو معامل ارتباط بيرسون Pearson's correlation coefficient.



لذلك، لا ينبغي استخدام التحليل التكراري المبسط للتشخيص السريري حيث لا يمكننا تحمل التنبؤات الخاطئة التي يمكن أن تضر بحياة الناس.

التعلم العميق البايزي على scRNAseq مع PyMC3

هنا سأستخدم بيانات scRNAseq عن الخلايا الليفية المرتبطة بالسرطان Cancer Associated Fibroblats (CAFs) وأطبق التعلم العميق البايزي لتصنيفها بين أنواع الخلايا الخبيثة malignant وغير الخبيثة non-malignant. بطريقة مماثلة، يمكن تصنيف مرضى السكري إلى أنواع فرعية من الأمراض للحصول على وصفة علاج دقيقة. سنبدأ بتنزيل بيانات التعبير من هنا، وتحميلهافي Python، وتقسيمها إلى مجموعات فرعية للتدريب والتحقق من الصحة والتصور باستخدام tSNE. كالعادة، صفوف مصفوفة التعبير عبارة عن عينات samples/خلايا cells، والأعمدة هي ميزات features/جينات genes، ويحتوي العمود الأخير على تسميات خلايا مشتقة من تجميع DBSCAN غير المتحيزة unbiased DBSCAN clustering.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.manifold import TSNE
from sklearn.decomposition import PCA
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split

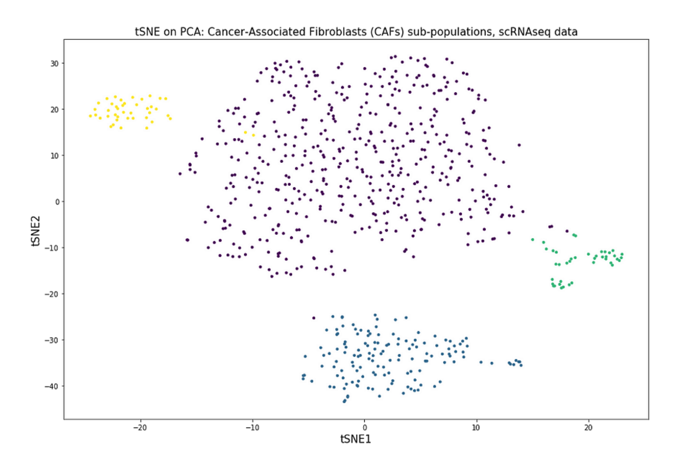
# READ AND LOG-TRANSFORM DATA
expr = pd.read_csv('expr_rpkm.txt', sep='\t')
X = expr.values[:,0:(expr.shape[1]-1)]
Y = expr.values[:,expr.shape[1]-1]
X = np.log(X + 1)

# PLOT TSNE ON PCA
X_reduced = PCA(n_components = 30).fit_transform(X)
model=TSNE(learning_rate=10,n_components=2,random_state=123,perplex
ity=30)
tsne = model.fit_transform(X_reduced)
```

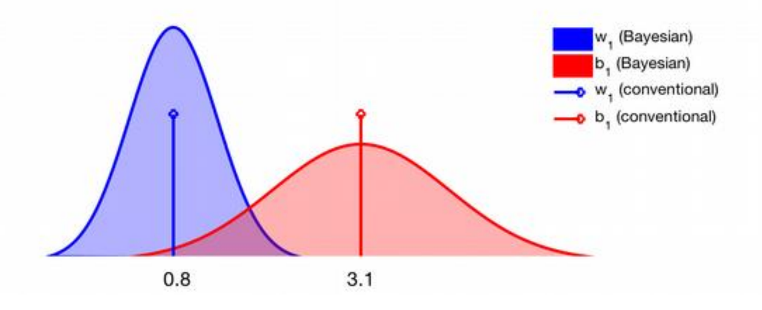
```
plt.scatter(tsne[:, 0], tsne[:, 1], c = Y, cmap = 'viridis', s =
10)
plt.title('tSNE: CAFs, scRNAseq data',fontsize = 15)
plt.xlabel("tSNE1", fontsize = 15); plt.ylabel("tSNE2", fontsize =
15)
plt.show()

# SPLIT DATA INTO TRAINING AND VALIDATION SUBSETS
X_train,X_test,Y_train,Y_test =
train_test_split(X_tsne,Y_tsne,test_size = 0.4,

stratify=None,random_state=12)
Y_train=np.array(OneHotEncoder().fit_transform(Y_train.reshape(-
1,1)).todense())
Y_test=np.array(OneHotEncoder().fit_transform(Y_test.reshape(-
1,1)).todense())
```



يمكن تمييز أربع مجموعات بوضوح في مخطط tSNE. بعد ذلك، سنقوم ببناء نموذج الشبكة العصبية البايزية Bayesian Neural Network (BNN) بطبقة واحدة مخفية و16 خلية عصبية، ويتم ذلك عن طريق تعيين Normal Priors للأوزان weights والتحيزات biases وتهيئتها بقيم عشوائية.



لبناء BNN، سأستخدم PyMC3 وأتبع النهج الموضحفي مدونة Thomas Wiecki الرائعة. ضمن النموذج، نحدد أيضًا الاحتمال الذي يمثل توزيعًا قاطعًا لأننا نتعامل مع مشكلة تصنيف scRNAseq متعددة الفئات (4 فئات classes).

```
import theano
import pymc3 as pm
import numpy as np
import theano.tensor as tt

def build_bayesian_neural_network(X, Y):
    n_samples = X.get_value().shape[0]
    n_classes = Y.get_value().shape[1]
    n_input = X.get_value().shape[1]
    n_hidden = 16
    with pm.Model() as model:
        # Priors for unknown model parameters

w_hidden=pm.Normal('w_hidden',mu=0,sd=1,shape=(n_input,n_hidden),
                              testval =
np.random.randn(n_input,n_hidden))

w_output=pm.Normal('w_output',mu=0,sd=1,shape=(n_hidden,n_classes),

testval=np.random.randn(n_hidden,n_classes))
        b_hidden=pm.Normal('b_hidden',mu=0,sd=1,shape = (n_hidden),
                              testval = np.random.randn(n_hidden))
        b_output=pm.Normal('b_output',mu=0,sd=1,shape=(n_classes),
                              testval=np.random.randn(n_classes))
        # Expected value of outcome

hidden_layer=pm.math.tanh(pm.math.dot(X.get_value(),w_hidden)+b_hid
den)
        output_layer=pm.Deterministic('output_layer',
                                         tt.nnet.softmax(

pm.math.dot(hidden_layer,w_output)
                                                          +
b_output))
        # Likelihood (sampling distribution) of observations
        likelihood = pm.Categorical('likelihood', output_layer,
                                     observed =
np.where(Y.get_value())[1])
    return model
```

من خلال وضع Priors على الأوزان والتحيزات، ندع النموذج يعرف أن هذه المعلمات بها uncertainties، وبالتالي فإن اخذ عينات MCMC (MCMC sampler) ستنشئ توزيعات لاحقة Posterior distributions لها. سنقوم الآن بتحديد دالة تستمد عينات من المؤثرات الخلفية لمعلمات شبكة BNN باستخدام أحد خوارزميات Hamiltonian Monte Carlo (عينة أسرع بكثير مقارنةً بـ Metropolis على سبيل المثال عندما يمكن حساب مشتقات المعلمات) خوارزميات تسمى NUTS. أخذ العينات هو تدريب BNN.

```
def train_bayesian_neural_network(model):

    with model:
        draws = 1000
        start = pm.find_MAP(maxeval = 10000)
        step = pm.NUTS()
        trace = pm.sample(draws=draws, step=step, start=start)

    return trace

X = theano.shared(X_train)
Y = theano.shared(Y_train)
model = build_bayesian_neural_network(X, Y)
trace = train_bayesian_neural_network(model)
```

سنقوم الآن بالتحقق من صحة تنبؤات نموذج الشبكة العصبية البايزية باستخدام إجراء الفحص التنبئي الخلفي Posterior Predictive Check (PPC). لهذا الغرض، سوف نستخدم النموذج المدرب ونرسم حدود القرار decision boundary على مخطط tNSE لمجموعة الاختبار الفرعية. يتم إنشاء حدود القرار عن طريق بناء شبكة 100 × 100 على مخطط tSNE وتشغيل تنبؤ النموذج لكل نقطة من الشبكة. بعد ذلك، نحسب المتوسط والانحراف المعياري لاحتمال تعيين كل نقطة على الشبكة إلى أحد الأنواع الفرعية للخلايا الأربعة وتصور متوسط الاحتمال mean probability وعدم اليقين uncertainty بشأن الاحتمال.

```
x_min, x_max = X_test[:, 0].min() - 1, X_test[:, 0].max() + 1
y_min, y_max = X_test[:, 1].min() - 1, X_test[:, 1].max() + 1
xx,yy=np.meshgrid(np.linspace(x_min,x_max,101),np.linspace(y_min,y_
max,101))
my_grid_values = np.c_[xx.ravel(), yy.ravel()]
dummy_out = np.array([[0,1,0,0]]*10201)

Z_prob_list = []
for k in range(10):
    X.set_value(np.float32(my_grid_values))
    Y.set_value(np.float32(dummy_out))
    model = build_bayesian_neural_network(X, Y)
    ppc = pm.sample_posterior_predictive(trace, model = model,
samples = 500)

    my_probs = []
    for j in range(ppc['likelihood'].shape[1]):
        my_probs.append([sum(np.array([i[j] for i in
ppc['likelihood']]) == 0) /
                         ppc['likelihood'].shape[0],
                         sum(np.array([i[j] for i in
ppc['likelihood']]) == 1) /
                         ppc['likelihood'].shape[0],
                         sum(np.array([i[j] for i in
ppc['likelihood']]) == 2) /
                         ppc['likelihood'].shape[0],
                         sum(np.array([i[j] for i in
ppc['likelihood']]) == 3) /
```

```
                          ppc['likelihood'].shape[0]])
    Z_prob = np.array([np.max(i) for i in np.array(my_probs)])
    Z_prob_list.append(Z_prob)

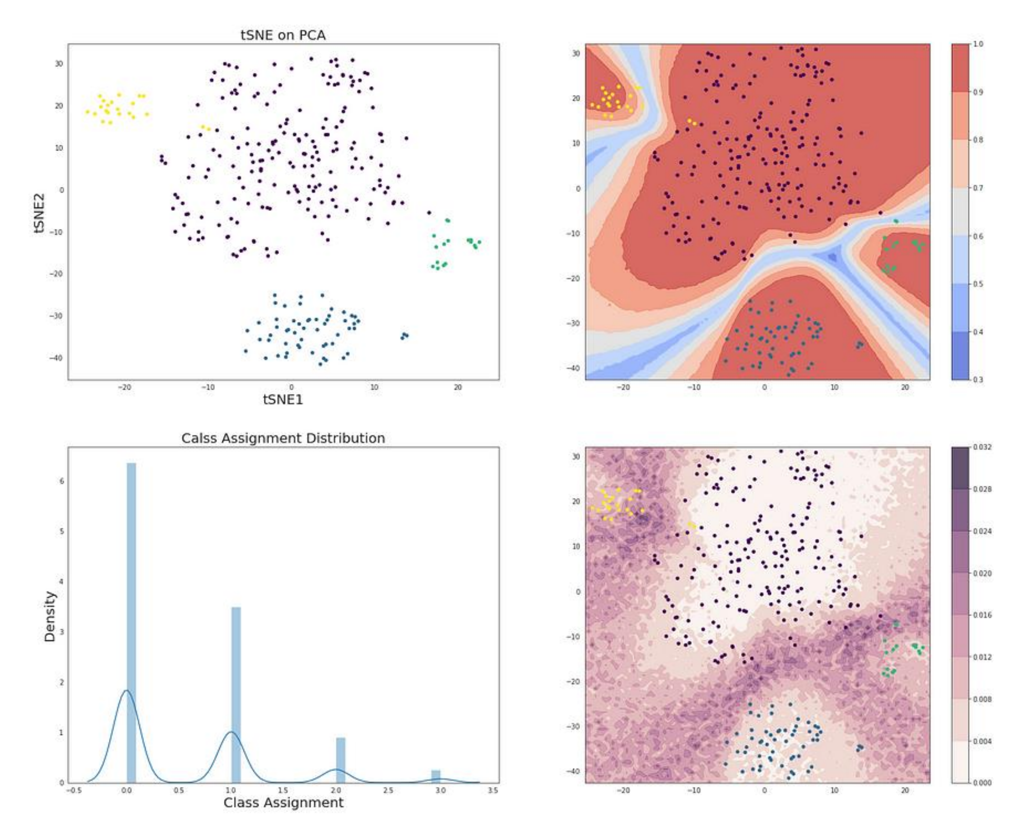
Z_prob_mean = np.array(Z_prob_list).mean(axis=0)
Z_prob_mean = Z_prob_mean.reshape(xx.shape)
Z_prob_std = np.array(Z_prob_list).std(axis=0)
Z_prob_std = Z_prob_std.reshape(xx.shape)
Z_prob_std

fig = plt.figure(figsize=(25, 20))
plt.subplot(221)
plt.scatter(X_test[:, 0], X_test[:, 1], c=Y_test_original,
cmap='viridis', s=20)
plt.title('tSNE on PCA', fontsize = 20)
plt.xlabel("tSNE1", fontsize = 20); plt.ylabel("tSNE2", fontsize =
20)

plt.subplot(222)
plt.contourf(xx, yy, Z_prob_mean, cmap = plt.cm.coolwarm, alpha =
0.8)
plt.colorbar()
plt.scatter(X_test[:, 0], X_test[:, 1], c=Y_test_original,
cmap='viridis', s=20)
plt.xlim(xx.min(), xx.max()); plt.ylim(yy.min(), yy.max())

plt.subplot(223)
sns.distplot(np.round(ppc['likelihood'].mean(axis=0)))
plt.title('Calss Assignment Distribution', fontsize = 20)
plt.xlabel("Class Assignment",fontsize=20);
plt.ylabel("Density",fontsize=20)

plt.subplot(224)
cmap = sns.cubehelix_palette(light=1, as_cmap=True)
plt.contourf(xx, yy, Z_prob_std, cmap = cmap, alpha = 0.8)
plt.colorbar()
plt.scatter(X_test[:, 0], X_test[:, 1], c=Y_test_original,
cmap='viridis', s=20)
plt.xlim(xx.min(), xx.max()); plt.ylim(yy.min(), yy.max())
plt.show()
```



حدود القرار الاحتمالية مع خريطة عدم اليقين لتصنيف CAFs

المخططات أعلاه تتوافق مع tSNEفي مجموعة الاختبار الفرعية (أعلى اليسار)؛ tSNEفي المجموعة الفرعية للاختبار مع متوسط احتمال تخصيص كل نقطة لأي من الأنواع الفرعية للخلايا الأربعة (أعلى اليمين)، وهو ما الاحتمال الأرجح / الشبكة العصبية التكرارية Maximum Likelihood Frequentist Neural Network؛ و tSNEفي مجموعة الاختبار الفرعية مع عدم اليقين من احتمال تخصيص كل نقطة لأنواع الخلايا الأربعة (أسفل اليمين)، وهو ناتج معين لشبكة Bayesian Neural Network. هنا تشير الألوان الحمراء والزرقاء إلى احتمالية عالية ومنخفضة لتعيين نقاط tSNE لأي نوع فرعي من الخلايا، على التوالي. تشير المنطقة المظلمةفي خريطة حرارة عدم اليقين إلى مناطق عالية عدم اليقين.

ما يمكننا رؤيته على الفور هو أن متوسط مخطط الحرارة الاحتمالي يحتوي على خليتين من الفئة الصفراء تم تعيينهما باحتمالية 100٪ للمجموعة الأرجواني. يعد هذا تصنيفًا خاطئًا للغاية وإثباتًا لفشل الاحتمال الأرجح / الشبكة العصبية المتكررة.في المقابل، تُظهر خريطة حرارة عدم اليقين أن الخليتين الصفراء تسقطانفي منطقة مظلمة نسبيًا مما يعني أن شبكة بايزي العصبية لم تكن متأكدة على الإطلاق

من تخصيص هذه الخلايا لأي مجموعة. يوضح هذا المثال قوة التعلم العميق البايزيفي إجراء تصنيفات أكثر أمانًا وأقل جذرية والتي لها أهمية خاصة للتشخيصات السريرية.

الاستنتاج

لقد تعلمنا هنا أن التعلم العميق البايزي هو طريقة أكثر دقة وأمانًا للقيام بالتنبؤات، وهو أمر منطقي جدًا لاستخدامهفي التشخيص السريري حيث لا يُسمح لنا أن نخطئفي وصفات العلاج. لقد استخدمنا PyMC3 وMCMC من أجل بناء نموذج التعلم العميق البايزي وعينة من الاحتمال اللاحق لتخصيص العينات للفئات الخبيثة مقابل غير الخبيثة. أخيرًا، أظهرنا تفوق التعلم العميق البايزي على النهج المتكررفي استخدام معلومات عدم اليقين لتجنب سوء تصنيف العينة.

المصدر:

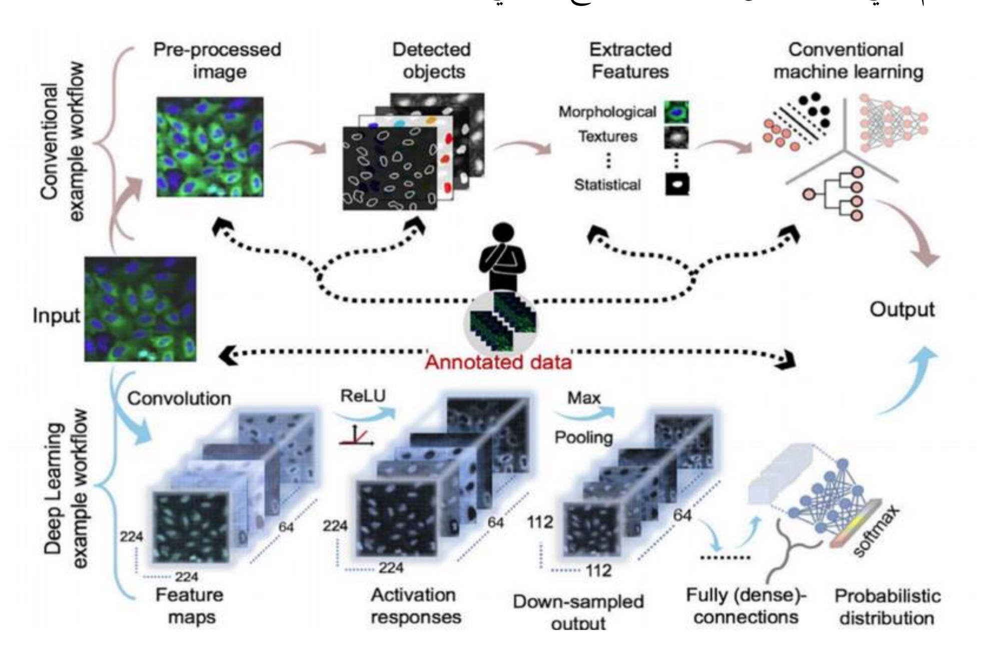
https://towardsdatascience.com/deep-learning-for-clinical-diagnostics-ca7bc254e5ac

12) التعلم العميق في التصوير المجهري Deep Learning on Microscopy Imaging

(كشف الخلايا الجيدة والسيئة والقبيحة باستخدام التعلم العميق)

هذه هي المقالة الخامسةفي سلسلة التعلم العميق لعلوم الحياة.في المنشورات السابقة، أوضحت كيفية استخدام التعلم العميق على الحمض النووي القديم والتعلم العميق لبيولوجيا الخلية المفردة والتعلم العميق لتكامل البيانات والتعلم العميق للتشخيص السريري. سنتحدث اليوم عن أحد التطبيقات الرئيسية للتعلم العميق لعلوم الحياة وهو رؤية الكمبيوتر Computer Vision لتحليل الصور المجهرية microscopy image analysis.

المجهر الآلي الفلوري Automated fluorescence microscopy هو حصان عامل للتصوير المجهري لعلوم الحياة الذي يولد ملايين الصور الخلوية cellular images. تتفوق الشبكات العصبية التلافيفية (CNNs)في تعلم البنية المكانية spatial structure لبيانات التصوير وتتفوق على أساليب التعلم الآلي التقليدية من خلال الاستخراج التلقائي للميزات automatic feature extraction.



سير العمل التقليدي مقابل التعلم العميق من A. Gupta et al.، Cytometry 95، 380366– (2019)

في حين أن تصنيف الصور بفئة واحدة لكل صورة (المعروف أيضًا باسم القطط مقابل الكلاب cats vs. dogs) يعد حاليًا مهمة تافهة جداً، إلا أن التصنيف الفعال متعدد التسميات multi-label

classification واكتشاف الكائنات يمثلان مشكلات أكثر صعوبة حيث يتم حاليًا تطوير جيل جديد من شبكات CNN.

سأصف هنا كيفية استخدام الشبكات العصبية الاصطناعية Faster-RCNN وMask-RCNN للكشف عن أنواع الخلايا على صور المجهر الفلوري fluorescence microscopy images من Human Protein Atlas (HPA).

الصورة مقابل البيانات الرقمية

عندما يتعلق الأمر بتحليل الصور image analysis، عادةً ما أتساءل عن سبب تفوق التعلم العميق (مقارنةً بالتعلم الآلي Machine Learning والتحليل الإحصائي statistical analysis)في العمل مع الصور وتطبيق أقل نجاحًا على البيانات الرقمية numeric data؟في الواقع، من خلال العمل مع بيانات Omics لتسلسل الجيل التالي الطبي الحيوي Next Generation Sequencing (NGS) (علم الجينوم genomics، والنسخ transcriptomics، والبروتيوميات proteomics، وما إلى ذلك)، والتي تكون عادةً رقمية، لا أرى غالبًا الشبكات العصبية neural networks تتفوق في الأداء على سبيل المثال على الغابة العشوائية Random Forest في القدرة التنبؤية، وبالتالي نوصي دائمًا بالبدء بنماذج خطية بسيطة قبل الغوص في الرياضيات غير الخطية باستخدام التعلم العميق.

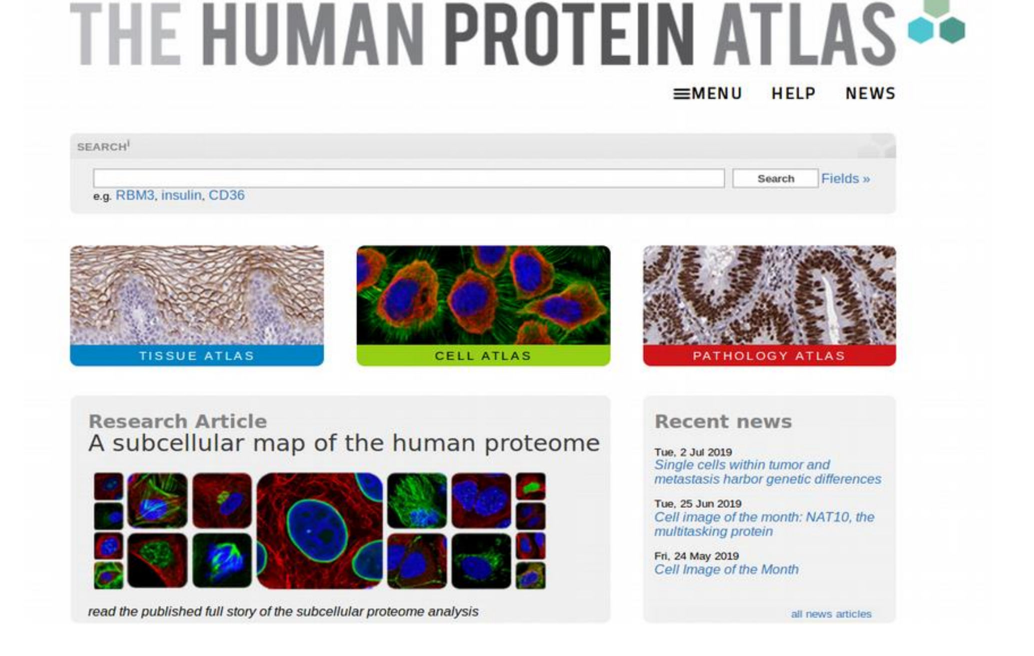
لماذا يعد التعلم العميق جيدًا لتحليل الصور؟

هل هو مجرد كمية هائلة من بيانات التصوير المتاحة كما يقترح جيفري هينتون Geoffrey Hintonأم أن هناك شيئًا غير خطي في شدة البكسل غير موجود في البيانات الرقمية؟ هل هي مجرد بُنية spatial structure مكانية للبيانات؟

أول شيء تلاحظه تشغيل التعلم العميق على بيانات الصورة هو أن الحياة تصبح أسهل مقارنة بإجراء التعلم العميق على بيانات Omics الرقمية. هناك الكثير من الأدبيات والعديد من البرامج التعليمية عبر الإنترنت التي يمكن أن تساعدك. أنا معجب كبير بجيسون براونلي Jason Brownlee وأدريان روزبروك Adrian Rosebrock، أذهب إلى مدوناتهم وعادة ما أجد إجابات على جميع أسئلتي.في المقابل، تشغيل التعلم العميق على البيانات الرقمية Omics (مثل تسلسل الحمض النووي الريبي RNA sequencing) أنت في الأساس بنفسك. لا يوجد الكثير من الأشخاص الذين يعرفون أفضل منك ما هي بُنية الشبكة network architecture ودالة التنشيط activation function والمحسن optimizer وما إلى ذلك التي تناسب بيانات Omics الرقمية الخاصة بك، لذلك تحتاج حقًا إلى تجربة الكثير.

التصوير المجهري للخلايا

ومع ذلك، عند دخولك في التصوير المجهري microscopy imaging ستجد الكثير من الدعم من المجتمع. أحد الموارد الرائعة المتاحة هو Human Protein Atlas (HPA) الذي يوفر، من بين أشياء أخرى ، بيانات صور رقمية توضح توطين localization البروتينات في خلايا مفردة.



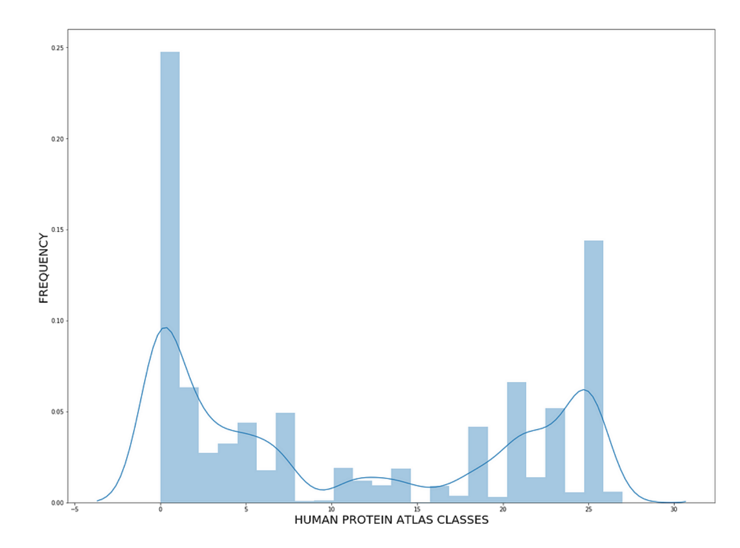
يعد Human Protein Atlas (HPA) مصدرًا رائعًا لبيانات الصور الرقمية

تم شرح البيانات باستخدام citizen science approach via online game واستخدامها في مسابقة Kaggle لتصنيف الصور متعدد التصنيفات multi-label image classification. يمكن تنزيل البيانات من هنا، وهي تشمل ما يقرب من 124000 صورة تدريب و 47000 صورة اختبار 512 x512 PNG لـ 28 فئة، أي أرقام من 0 إلى 27 رمزًا لمقصورات الخلايا cell compartments حيث يتم التعبير عن بروتين معين. الأهم من ذلك، يمكن أن توجد فئات متعددة على نفس الصورة حيث يمكن التعبير عن البروتينات في عدة أماكن من الخلية في وقت واحد.

```
name_label_dict = {
0:  'Nucleoplasm',
1:  'Nuclear membrane',
2:  'Nucleoli',
3:  'Nucleoli fibrillar center',
4:  'Nuclear speckles',
5:  'Nuclear bodies',
6:  'Endoplasmic reticulum',
7:  'Golgi apparatus',
```

```
8:  'Peroxisomes',
9:  'Endosomes',
10:  'Lysosomes',
11:  'Intermediate filaments',
12:  'Actin filaments',
13:  'Focal adhesion sites',
14:  'Microtubules',
15:  'Microtubule ends',
16:  'Cytokinetic bridge',
17:  'Mitotic spindle',
18:  'Microtubule organizing center',
19:  'Centrosome',
20:  'Lipid droplets',
21:  'Plasma membrane',
22:  'Cell junctions',
23:  'Mitochondria',
24:  'Aggresome',
25:  'Cytosol',
26:  'Cytoplasmic bodies',
27:  'Rods & rings' }
```

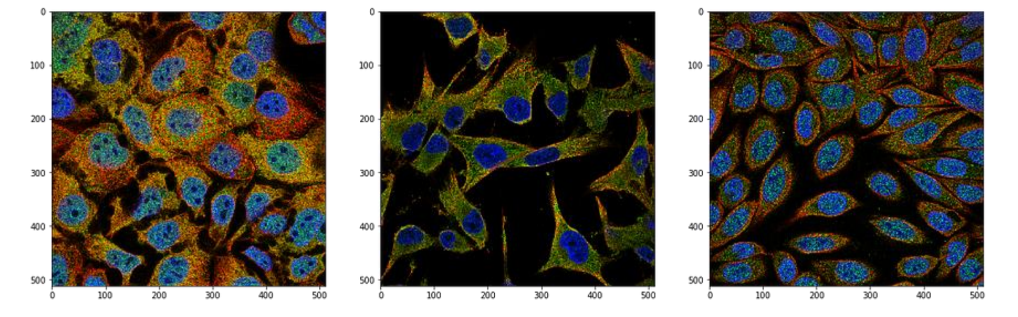
```
import numpy as np
import pandas as pd
import seaborn as sns
annot = pd.read_csv('train.csv')
my_classes = [i.split(' ') for i in annot['Target'].values]
my_classes = [int(y) for x in my_classes for y in x]
sns.distplot(my_classes)
plt.xlabel('HUMAN PROTEIN ATLAS CLASSES', fontsize = 20)
plt.ylabel('FREQUENCY', fontsize = 20)
```



بالنظر إلى توزيع فئات HPA، يمكننا أن نرى أن البروتينات ذات الأهمية يتم التعبير عنها غالبًافي بلازم نووي Nucleoplasm (الفئة 0) وعصارة الخلية Cytosol (الفئة 25). الآن دعونا نتحقق من شكل

صور HPA. اتضح أن عرض صور HPA هي مهمة غير تافهة لأنها تحتوي على 4 قنوات بدلاً من 3 قنوات RGB القياسية التي تسلط الضوء على البروتين المهم (القناة الخضراء green channel) بالإضافة إلى ثلاثة معالم خلوية cellular landmarks: الأنابيب الدقيقة microtubules (الحمراء) والنواة nucleus (الزرقاء) والشبكة الإندوبلازمية endoplasmic reticulum (أصفر). تتمثل إحدى طرق دمج القنوات الأربعةفي Pythonفي إدراك أن اللون الأصفر = أحمر + أخضر وإضافة نصف القناة الصفراء إلى الأحمر والنصف الآخر إلى القناة الخضراء.

```
from PIL import Image
import matplotlib.pyplot as plt
def load_image(path, id):
    R = np.array(Image.open(path + id + '_red.png'))
    G = np.array(Image.open(path + id + '_green.png'))
    B = np.array(Image.open(path + id + '_blue.png'))
    Y = np.array(Image.open(path + id + '_yellow.png'))
    image = np.stack((R + Y/2, G + Y/2, B),-1).astype(np.float32)
    return image
my_image = load_image('/HPA/','0a1fe790-bac8-11e8-b2b7-
ac1f6b6435d0')
plt.imshow((my_image).astype(np.int))
```

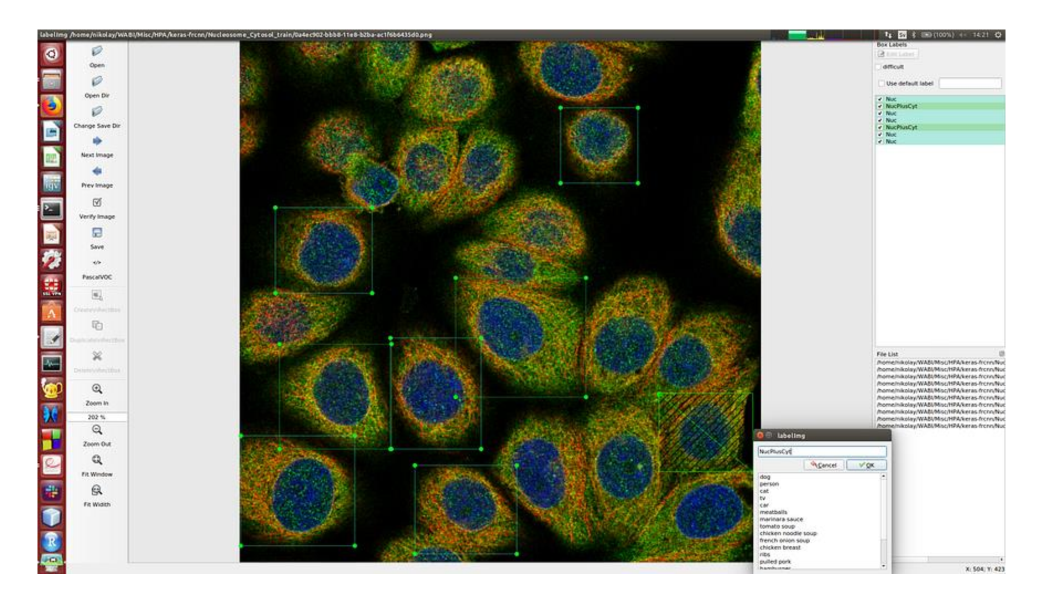


باستخدام دالة "load_image"، قمت بإنشاء مجموعة بيانات تدريب جديدة تضم حوالي 31000 صورة عن طريق دمج 4 قنوات لكل معرف صورة.

بناء تعليق توضيحي لاكتشاف الخلية

الآن بعد أن تعلمنا كيفية عرض صور HPA ودمج القنوات الأربع، حان الوقت لإنشاء تعليقات توضيحية annotations لاكتشاف نوع الخلية مع الشبكات العصبية Faster-RCNN و Mask-RCNN. لأغراض العرض، قمت بمراجعة عدد قليل من الصور والصور المختارة التي تحتوي على كلتا الخلايا مع البروتين المعبر عنهفي مقصورات Nucleoli (Nucleoli compartments) (Nucleoli, Nucleoli fibrillar center, Nuclear speckles, Nuclear bodies) وخلايا بدون علامات على تعبير البروتينفي مقصورات Nucleoli. وبهذه الطريقة، هدفت إلى الحصول على 3 فئات (Nucleoli وليس Nucleoli وخلفية background)في كل صورة من صور التدريب /

الاختبار الخاصة بي. بعد ذلك، أمضيت ساعتين مع LabelImg لتعيين مربعات bounding boxes إحاطة وتسميات فئة لكل خلية من أجل 45 صورة قطار HPA، تم حجز 5 صور أخرى كمجموعة بيانات اختبار لاستخدامهافي إجراء التنبؤات.



التعليق التوضيحي اليدوي للخلايا باستخدام LabelImg

يقوم LabelImg بتسجيل التعليقات التوضيحية للخلايا بتنسيق xml الذي يحتوي على البنية النموذجية التالية:

```
<annotation>
        <folder>train</folder>
        <filename>0a3028e6-bbca-11e8-b2bc-
ac1f6b6435d0.png</filename>
        <path>/HPA/0a3028e6-bbca-11e8-b2bc-ac1f6b6435d0.png</path>
        <source>
                <database>Unknown</database>
        </source>
        <size>
                <width>512</width>
                <height>512</height>
                <depth>3</depth>
        </size>
        <segmented>0</segmented>
        <object>
                <name>Nucleoli</name>
                <pose>Unspecified</pose>
                <truncated>0</truncated>
                <difficult>0</difficult>
                <bndbox>
                        <xmin>317</xmin>
```

```
                        <ymin>410</ymin>
                        <xmax>411</xmax>
                        <ymax>507</ymax>
                </bndbox>
        </object>
</annotation>
```

هنا يمكنك رؤية العرض (512 بكسل)، الارتفاع (512 بكسل) والعمق (3 قنوات) للصورة، وكائن واحد بإحداثيات محددة بواسطة مربع الاحاطة (xmin، ymin، xmax، ymax) والتسمية "Nucleoli ". يحتاج Faster-RCNN إلى ملف تعليق توضيحي بتنسيق خاص مفصول بفواصل (csv). لإعداد ملف csv هذا، نحتاج إلى تحليل تعليقات xml التوضيحية لكل صورة:

```
def build_annot(file):
    from xml.etree import ElementTree
    tree = ElementTree.parse(file)
    root = tree.getroot()
    x_min = list()
    x_max = list()
    y_min = list()
    y_max = list()
    for box in root.findall('.//bndbox'):
        x_min.append(int(box.find('xmin').text))
        y_min.append(int(box.find('ymin').text))
        x_max.append(int(box.find('xmax').text))
        y_max.append(int(box.find('ymax').text))
    my_classes = list()
    my_images = list()
    for name in root.findall('.//object'):
        my_classes.append(name.find('name').text)
        my_images.append('train/' +
root.findall('.//filename')[0].text)

    annot_df = pd.DataFrame({'Image': my_images,'xmin': x_min,
'xmax': x_max,
                             'ymin': y_min, 'ymax': y_max,
                             'Class': my_classes})
    return annot_df

import os
annot_df = pd.DataFrame()
for i in os.listdir('/HPA/train_annot/'):
    annot_df = pd.concat([annot_df, build_annot('/HPA/train_annot/'
                                               + str(i))],
ignore_index=True)
annot_df.to_csv('/HPA/annot.txt', header=False, index=False,
sep=',')
```

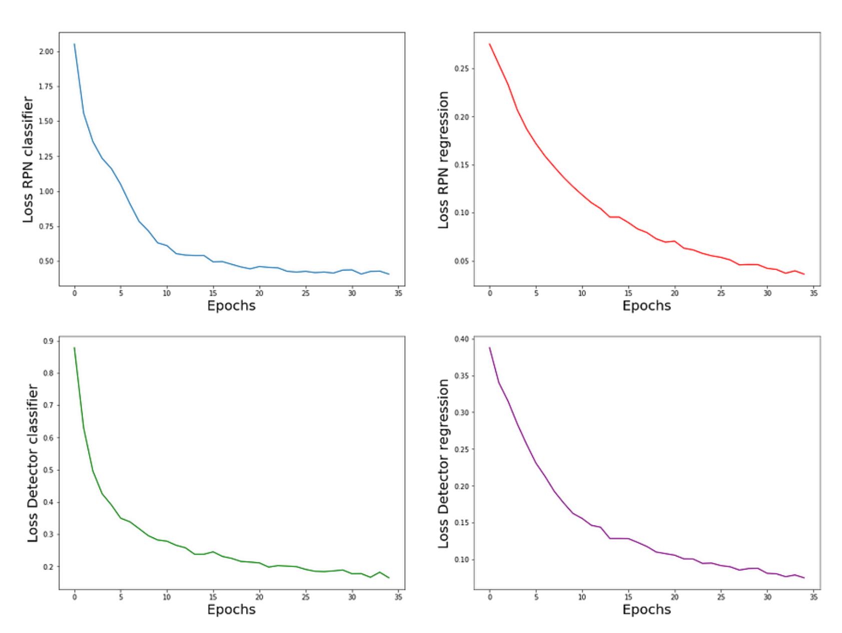
بمجرد تحليل تعليقات xml، تكون قد انتهيت تقريبًا، والآن أصبح كل شيء جاهزًا لتدريب نموذج Faster-RCNN.

تدريب Faster-RCNN لاكتشاف الخلايا

لقد تخطيت هنا شرح كيفية عمل Faster-RCNN، فهناك الكثير من الأدبيات التي تتناول الخوارزمية. أذكر فقط أن Faster-RCNN تستخدم شبكة اقتراح المنطقة Region Proposal Network (RPN) التي تنشئ مقترحات المنطقة لشبكة الكاشف Detector التي تقوم باكتشاف الكائن الفعلي. ومن ثم، فإن دالة الخطأ loss function لشبكة Faster-RCNN تجمع بين المساهمات من الانحدار regression (تحديد الخلايا مع مربعات الاحاطة) ومهام التصنيف classification (تعيين فئة لكل خلية المحددة localized cell). يمكن تثبيت Faster-RCNN من https://github.com/kbardool/keras-frcnn. يعد تدريب نموذج Faster-RCNN سهلاً مثل الكتابة:

```
python train_frcnn.py -o simple -p annot.txt --hf --vf --rot --
num_epochs 500
```

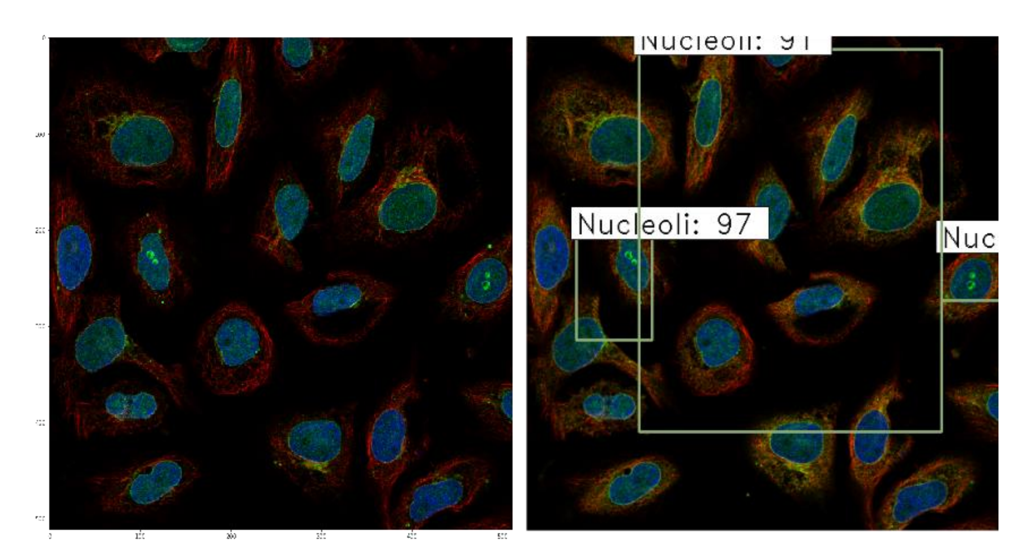
هنا "annot.txt" هو ملف التعليق التوضيحي الذي تم إنشاؤهفي القسم السابق. بشكل افتراضي، يستخدم Faster-RCNN نقل التعلم باستخدام أوزان من ResNet50. استغرق تدريب Faster-RCNN على 45 صورة مع 322 من فئات Nucleoli المشروحة و 306 Not Nucleoli ما يقرب من 6 ساعات لكل فترة epoch على الكمبيوتر المحمول الخاص بي باستخدام 4 نوى لوحدة المعالجة المركزية، لذلك تمكنت فقط من الانتظار لمدة 35 فترة. يبدو أن منحنيات التعلم توضح أن مهمة التصنيف وصلت إلى التشبع saturation بينما كان الانحدار (تحديد الخلايا) بعيداً عن الوصول إلى الهضبة plateau.



لعمل تنبؤات حول مجموعة الاختبار باستخدام Faster-RCNN، نكتب ببساطة:

```
python test_frcnn.py -p test
```

هنا "اختبار" test هو المجلد الذي يحتوي على الصور من مجموعة بيانات الاختبار. دعونا نعرض صورة اختبارية للتحقق من مدى نجاح النموذجفي اكتشاف الخلايا بالبروتين المعبر عنهفي Nucleoli (فئة Nucleoli) والخلايا التي لا تحتوي على البروتين المعبر عنهفي Nucleoli (ليس فئة Nucleoli):



الصورة الأصلية (يسار) والصورة بعد تطبيق اكتشاف كائن Faster-RCNN (يمين)

هنا، على اليسار صورة الاختبار الأصلية. من الواضح أن خليتين تحتويان على بقع خضراء زاهيةفي الوسط، تلك هي النوى التي تظهر بسبب البروتين محل الاهتمام الذي يظهر تعبيرًا قويًافي هذه المناطق، لذلك يجب أن تنتمي هاتان الخليتان إلى فئة Nucleoli. لا يبدو أن باقي الخلايا تحتوي على البروتين المعبر عنهفي مناطق Nucleoli، لذا يجب أن تنتمي إلى فئة Not Nucleoli. على اليمين توجد صورة الاختبار مع المربعات المحيطة وتسميات الفئة المحددة بواسطة نموذج Faster-RCNN المدرب. هنا يمكننا أن نرى أن النموذج اكتشف بشكل صحيح خليتين بهما بقع خضراء مرئيةفي مناطق Nucleoli بينما يبدو أن المربع المحيط الثالث هو تنبؤ إيجابي خاطئ false positive prediction، وليس من الواضح بالضبط ما الذي اكتشفه النموذج بثقة عالية (احتمال 91 ٪): يشتمل مربع الاحاطة على خلايا متعددة وعلى الرغم من وضع تسمية فئة "Nucleoli"في النموذج، فإننا لا نلاحظ الخلايا ذات النوى المرئية داخل مربع الاحاطة.

توضح هذه الصورة انطباعي العام حول استخدام Faster-RCNN لاكتشاف الخلايا: فهي ليست دائمًا مثاليةفي تحديد الخلية cell localization، ولكن قد يؤدي المزيد من التدريب إلى تحسينها. دعونا نرى ما إذا كان Mask-RCNN يمكنه تحسينه.

تدريب Mask-RCNN لاكتشاف الخلايا

ينتمي Mask-RCNN (بالإضافة إلى Faster-RCNN) إلى عائلة RCNN من الشبكات العصبية الاصطناعية المعروفة بدقة أعلىفي اكتشاف الكائنات object detection مقارنة بالعائلات الأخرى، مثل YOLO وSSD، والتي لا أغطيها هنا. بالإضافة إلى اكتشاف الكائن، يسمح Mask-RCNN أيضًا بتجزئة الكائن object segmentation، لكننا لن نستخدمه هنا. يمكن تثبيت Mask-RCNN من https://github.com/matterport/Mask_RCNN. بينما من السهل جدًا تشغيل Faster-RCNN (يحتاج بشكل أساسي إلى إعداد ملف التعليقات التوضيحية annotation file فقط)، يتطلب Mask-RCNN المزيد من البرمجة، لذا يرجى التحقق من نوتبوك Jupyter الكامل على github الخاص بي للحصول على مزيد من التفاصيل. أشرح هنا الخطوات الرئيسية لسير العمل الذي يتبع بشكل أساسي هذا البرنامج التعليمي الممتاز. تتمثل خصوصية Mask-RCNNفي أن البيانات تتم معالجتها بواسطة كائن مجموعة البيانات الذي سيتم استخدامه لتزويدهفي Mask-RCNN للتدريب والاختبار.

```
from os import listdir
from numpy import zeros
from numpy import asarray
from mrcnn.utils import Dataset

class HPADataset(Dataset):

    def load_dataset(self, dataset_dir, is_train=True):
        self.add_class("dataset", 1, "Nucleoli")
        self.add_class("dataset", 2, "Not Nucleoli")
        annotations_dir = dataset_dir + 'train_annot/'
        images_dir = dataset_dir + 'train/'
        for i,filename in
zip(range(len(listdir(images_dir))),listdir(images_dir)):
            image_id = filename[:-4]
            if is_train and i >= 40:
                continue
            if not is_train and i < 40:
                continue
            img_path = images_dir + filename
            ann_path = annotations_dir + image_id + '.xml'
            self.add_image('dataset', image_id=image_id,
path=img_path,
                           annotation=ann_path)

    def load_mask(self, image_id):
        info = self.image_info[image_id]
```

```
        path = info['annotation']
        boxes, w, h, my_classes = extract_boxes(path)
        masks = zeros([h, w, len(boxes)], dtype='uint8')
        class_ids = list()
        for i in range(len(boxes)):
            box = boxes[i]
            row_s, row_e = box[1], box[3]
            col_s, col_e = box[0], box[2]
            masks[row_s:row_e, col_s:col_e, i] = 1
            if my_classes[i]=='Nucleoli':

class_ids.append(self.class_names.index('Nucleoli'))
            else:
                class_ids.append(self.class_names.index('Not
Nucleoli'))
        return masks, asarray(class_ids, dtype='int32')

    def image_reference(self, image_id):
        info = self.image_info[image_id]
        return info['path']

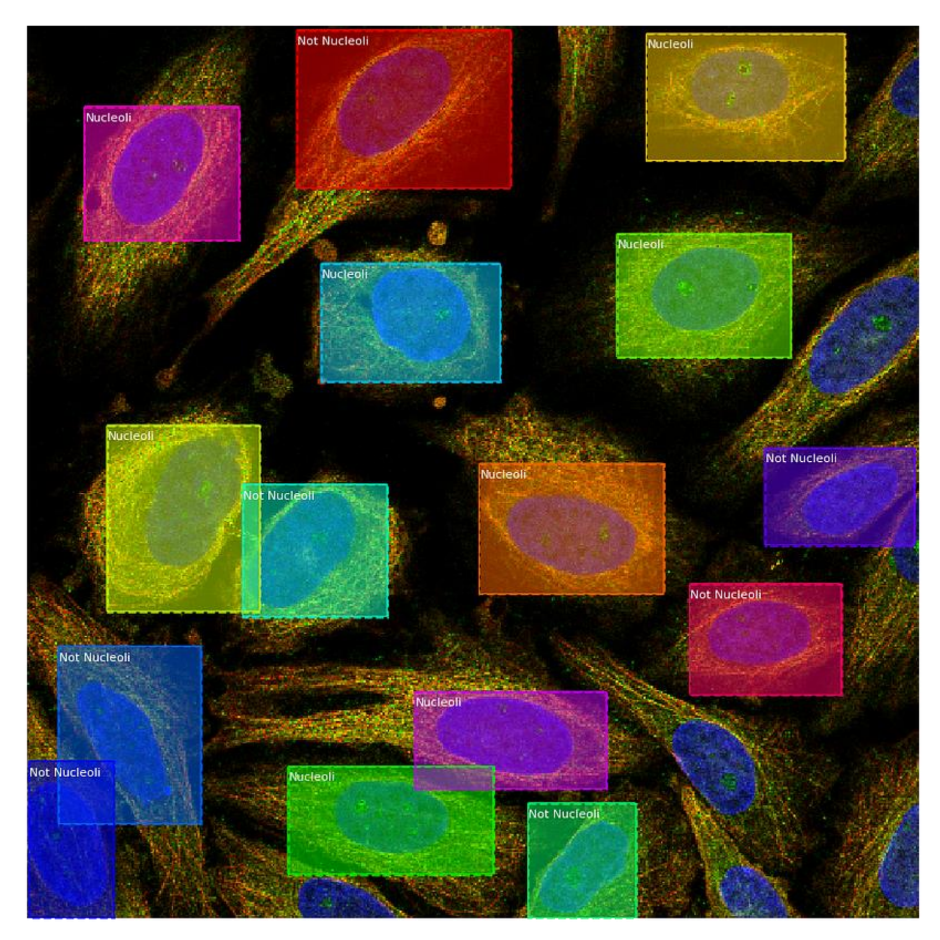
train_set = HPADataset()
train_set.load_dataset('/HPA/', is_train=True)
train_set.prepare()
print('Train: %d' % len(train_set.image_ids))

test_set = HPADataset()
test_set.load_dataset('/HPA/', is_train=False)
test_set.prepare()
print('Test: %d' % len(test_set.image_ids))
```

نحن هنا نهدف إلى اكتشاف الكائن بدلاً من التجزئة، لذلك سنتعامل مع مربعات الاحاطة كأقنعة masks، لذلك سيتم تحميل دالة "load_mask" في الواقع إحداثيات مربعات الاحاطة. يمكننا عرض صورة تدريب مشروحة عشوائية باستخدام دالة "display_instances" العملية من Mask-RCNN:

```
from mrcnn.visualize import display_instances
from mrcnn.utils import extract_bboxes

image_id = 5
image = train_set.load_image(image_id)
mask, class_ids, _ = train_set.load_mask(image_id)
bbox = extract_bboxes(mask)
display_instances(image, bbox, mask, class_ids,
train_set.class_names)
```



تم إعداد الصورة المشروحة للتدريب باستخدام Mask-RCNN

الآن كل شيء جاهز لتدريب نموذج Mask-RCNN. سنستخدم نقل التعلم transfer learning ونبدأ من الأوزان من نموذج Mask-RCNN لاكتشاف الكائنات المدربة مسبقًا pre-trained object detection على مجموعة بيانات COCO.

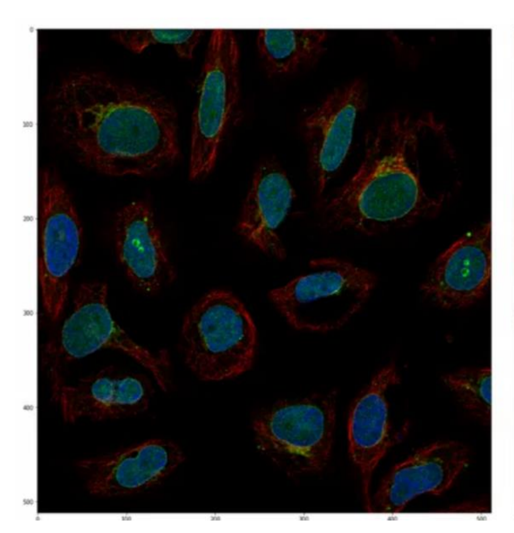
```
from mrcnn.config import Config
from mrcnn.model import MaskRCNN

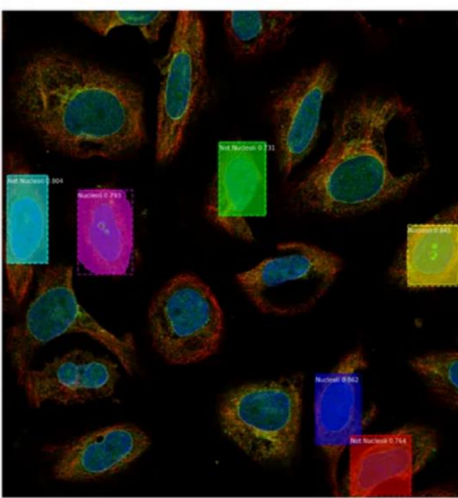
class HPAConfig(Config):
    NAME = "hpa_cfg"
    # Number of classes (background + Nucleoli + Not Nucleoli)
    NUM_CLASSES = 1 + 1 + 1
    # Number of training steps per epoch: number of images in the
training set
    STEPS_PER_EPOCH = 40

config = HPAConfig()
```

```
model = MaskRCNN(mode='training', model_dir='/HPA/', config=config)
model.load_weights('/HPA/mask_rcnn_coco.h5', by_name=True,
                   exclude=["mrcnn_class_logits", "mrcnn_bbox_fc",
                            "mrcnn_bbox", "mrcnn_mask"])
model.train(train_set, test_set,
learning_rate=config.LEARNING_RATE,
            epochs=5, layers='heads')
```

كان تدريب Mask-RCNN أسرع بشكل كبير مقارنةً بـ Faster-RCNN، استغرقت فترة واحدة ساعتين فقط (X3 سرعة مقارنة بـ Faster-RCNN) على الكمبيوتر المحمول الخاص بي مع 4 نوى من وحدة المعالجة المركزية، وتوقفت عن التدريب بعد 5 فترات فقط لأن نتائج الكائن كان الاكتشاف في مجموعة بيانات الاختبار بالفعل أكثر من مرضٍ. هنا للمقارنة، أقدم صورة الاختبار الأصلية (على اليسار) والصورة مع الثقة العالية (أكثر من 70٪ احتمال) مربعات الاحاطة وتسميات الفئات الموضوعة بواسطة نموذج Mask-RCNN المدرب.





الصورة الأصلية (يسار) والصورة بعد تطبيق اكتشاف كائن Mask-RCNN (يمين)

نلاحظ زيادة مذهلة في دقة تحديد الخلايا، ويبدو أن جميع مربعات الاحاطة تحتضن الخلايا بشكل مثالي تقريبًا. يبدو أن إحدى الخلايا تحمل تسمية خاطئة "Nucleoli" على الرغم من عدم وجود بقع خضراء واضحة يمكن ملاحظتها داخل النواة. ربما سيستفيد النموذج من المزيد من التدريب. بشكل عام، يُظهر Mask-RCNN تحسنًا ملحوظًا في اكتشاف الخلايا مقارنةً بـ Faster-RCNN.

يمكن الآن استخدام نموذج Mask-RCNN المدرب هذا لإجراء مسح ضوئي عالي الإنتاجية high-throughput للصور من أجل التشكل الخلوي الخاص special cellular morphology (مع البروتين المعبر عنه في مناطق Nucleoli في هذه الحالة بالذات) دون فحص بصري visual inspection.

الاستنتاج

في هذا المنشور، تعلمنا أن الفحص المجهري الآلي ينتج كميات كبيرة من بيانات الصور الرقمية التي تعتبر مناسبة بشكل مثالي للتحليل باستخدام التعلم العميق. يعد اكتشاف الأشكال الخلوية cellular morphologies مهمة صعبة على الرغم من توفر الكثير من الادبيات والنماذج. اختبرنا نماذج اكتشاف الكائنات Faster-RCNN و Mask-RCNN باستخدام صور مشروحة بفئات متعددة من Human Protein Atlas (HPA). تفوق Mask-RCNNفي الأداء على Faster-RCNN من حيث جودة وسرعة اكتشاف نوع الخلية.

المصدر:

https://towardsdatascience.com/deep-learning-on-microscopy-imaging-865b521ec47c

13) التعلم العميق على جينات الإنسان البدائي Deep Learning on Neanderthal Genes

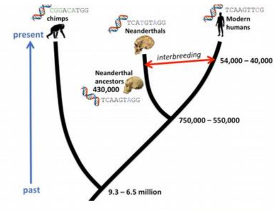
(الكشف عن مناطق أصل إنسان نياندرتال من خلال التعلم العميق)

هذا هو المنشور السابع من عمودي التعلم العميق لعلوم الحياة حيث أقدم أمثلة ملموسة عن كيفية تطبيق التعلم العميق الآنفي علم الأحياء الحسابي Computational Biology وعلم الوراثة Genetics والمعلوماتية الحيوية Bioinformatics.في المنشورات السابقة، أوضحت كيفية استخدام التعلم العميق للحمض النووي القديم Deep Learning for Ancient DNA، وبيولوجيا الخلية المفردة Single Cell Biology، وتكامل بيانات OMICs Data Integration OMIC، والتشخيص السريري Clinical Diagnostics والتصوير المجهري Microscopy Imaging. سنغوص اليومفي التاريخ المثير للتطور البشري History of Human Evolution ونتعلم أنه من السهل استعارة المنهجية من معالجة اللغة الطبيعية (NLP) Natural Language Processing وتطبيقها على علم الوراثة السكانية البشرية من أجل استنتاج مناطق تدخل الإنسان البدائي Neanderthal introgressionفي الجينوم البشري الحديث.

تاريخ موجز: خارج أفريقيا

عندما هاجر أسلاف البشر المعاصرين من إفريقيا ما بين 50000 و70000 سنة، واجهوا إنسان نياندرتال Neanderthals ودينيسوفان Denisovans، وهما مجموعتان من أشباه البشر القدامى الذين سكنوا أوروبا وآسيافي ذلك الوقت. نحن نعلم أن البشر المعاصرين تزاوجوا مع كل من إنسان نياندرتال ودينيسوفان نظرًا لوجود دليل على وجود الحمض النووي الخاص بهمفي جينومات البشر المعاصرين من أصل غير أفريقي. أصبحت هذه التطورات الجينية العظيمةفي تاريخ التطور البشري ممكنة بسبب تسلسل جينومات إنسان نياندرتال ودينيسوفانفي عامي 2010 و 2012، على التوالي، من قبل مجموعة Svante Pääbo.



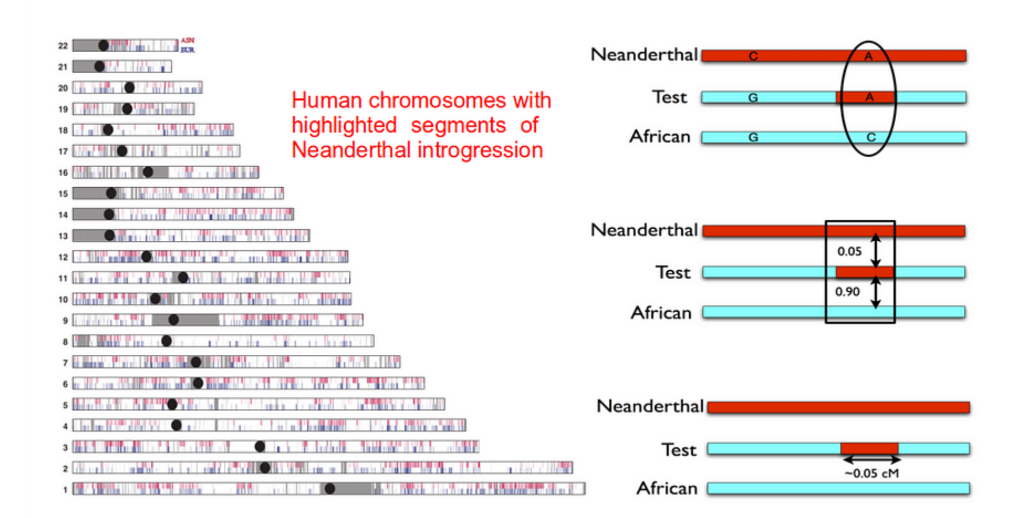






2010 Draft Neanderthal Genome Sequenced

كانت هناك محاولات قليلة لاستنتاج المواقع الدقيقة لشرائح الحمض النووي الموروثة من إنسان نياندرتال ودينيسوفان. لقد استخدموا موارد الجينوم المختلفة مثل مشروع 1000 جينوم وحسبوا مقاييس مختلفة للتشابه المحلي لجينوم غير أفريقي تجاه جينوم الإنسان البدائي عالي الجودة والتباعد فيما يتعلق بالجينومات الأفريقية (الشكل الأيمن أدناه).

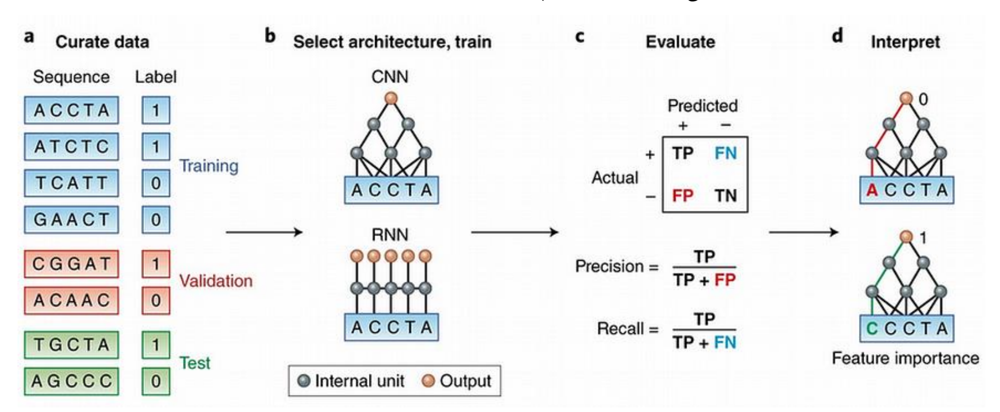


مقاييس التشابه هذه (إحصاءات موجزة) على الرغم من كونها قابلة للتفسير وفعالة تؤدي إلى فقدان المعلومات لأنها تحاول التقاط جزء من الحمض النوويفي رقم واحد دون النظرفي التسلسل المرتب للنيوكليوتيدات نفسها. تستخدم المحاولات الأخرى لمسح إدخال الإنسان البدائي عبر الجينوم نموذج ماركوف المخفي Hidden Markov Model (HMM) وهو نموذج بلا ذاكرة لا يفسر مرة أخرى الارتباطات طويلة المدى بين النيوكليوتيدات على طول تسلسل الحمض النووي. هذا هو المكان الذي يمكن أن تكون فيه قوة التعلم العميق لمعالجة التسلسلات الجينومية الأولية والاستفادة من الذاكرة الطويلة لرابط النوكليوتيدات عبر الجينوم مع RNNs / LSTMs وCNNs مفيدة بشكل لا يصدق.

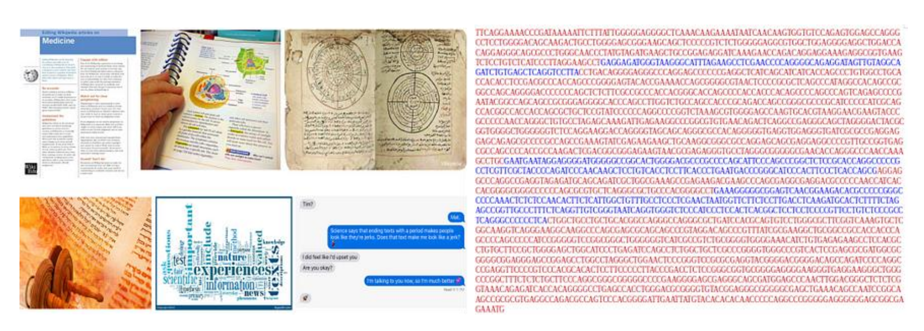
إمكانات التعلم العميق لعلم الجينوم القديم

في الوقت الحالي، يتم تجاهل التعلم العميق بشكل أساسيفي علم الأحياء التطوري Evolutionary Biology على الرغم من إمكاناته الكبيرة للعمل مع بيانات الجينوم التي تمثل أحد مصادر البيانات الرئيسيةفي مجالات العلوم التطورية الحالية وأبحاث الحمض النووي القديم. تستخدم المحاولات الأخيرة بيانات الجينوم المحاكية simulated genomic data على الرغم من توافر بيانات الجينوم الحقيقية التي غالبًا ما تحتوي على مناطق مشروحة بوظيفة بيولوجية مفهومة جيداً. التعلم العميق مناسب بشكل مثالي للعمل مع علم الجينوم Genomics وعلم الجينوم القديم Ancient Genomics لمجرد أن جينوم واحد هوفي الواقع بيانات كبيرة one genome is actually a Big

Data. لإدراك ذلك، فكر فقطفي تقطيع الجينوم الطويل 3 * 10^9 إلى امتدادات stretches من 1000 نيوكليوتيد والتي تجلب أمثلة تدريبية 3 * 10^6 لاستخدامهافي التعلم العميق بشرط أن يكون التعليق التوضيحي (التسميات labels) يمكن اكتشافه لكل امتداد من امتدادات الحمض النووي DNA stretches. هذه كمية هائلة من بيانات التدريب!

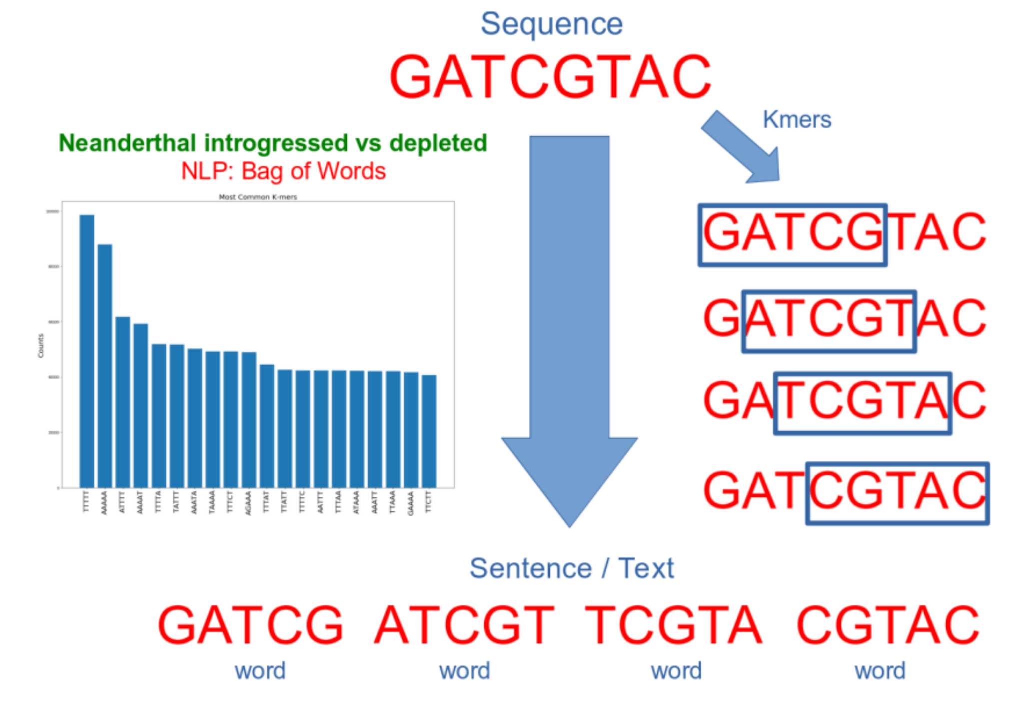


لقد أوضحت كيفية استخدام هذه الفكرة للتمييز بين تسلسل الحمض النووي القديم والحديثفي مقالتي السابقة. هناك استخدمت شبكة عصبية تلافيفية أحادية الأبعاد 1D Convolutional Neural Network (CNN) وتمثيلًا مشفرًا واحدًا ساخنًا one-hot-encoded للتسلسلات. سأقوم هنا بتوضيح طريقة أخرى لتمثيل تسلسل الحمض النووي للإدخالفي التعلم العميق. يرجى إلقاء نظرة على الشكل الأيسر أدناه، وهذا ما نعنيه عادة بالنص text. ومع ذلك، ليس هذا ما يقصده الناسفي علم الجينوم بالنص. يظهر ما يقصدونه بالنص إلى اليمين. تسلسل الحمض النووي هو نص! يبدو الأمر مملًا ولكن مع مرور الوقت تبدأفي رؤية الأشياءفي سلاسل نصية strings. ماذا لو أخبرتك أنني أرى جينًا gene هناك؟ ماذا لو أخبرتك أنني رأيت جينًا مرتبطًا بقوة بمرض السكري من النوع 2 Type 2 Diabetes (T2D) ومن المحتمل أننا ورثنا هذا الجين من إنسان نياندرتال؟



ماذا يقصد الناس عادة بالنص (يسار)، مقابل ما يقصده علماء المعلومات الحيوية بالنص (يمين)، جين SLC16A11

الآن، إذا كان تسلسل الحمض النووي عبارة عن نص، فيمكننا تطبيق جميع الأجهزة من معالجة اللغة الطبيعية (NLP) على مثل هذه النصوص. ومع ذلك، أين الجمل في نصوص الحمض النووي وأين الكلمات؟ إذا توقعنا ما إذا كانت مجموعة من تسلسلات الحمض النووي (يمكن اعتبار المجموعة bunch نصًا) موروثة من إنسان نياندرتال أم لا، فيمكن اعتبار تسلسل DNA واحد من المجموعة جملة من النص، و k-mer (التسلسل الفرعي sub-sequence) ككلمة.



تسلسل الحمض النووي عبارة عن جملة يمكن تقسيمها إلى k-mers، ويمكن اعتبار k-mers محددة المسافات ككلمات

بمجرد تحويل تسلسلات الحمض النووي إلى k-mers محددة بمسافة / كلمات space-delimited words / k-mers، نكون قد انتهينا. يمكننا الآن اللجوء إلى تقنيات المعالجة اللغوية الطبيعية المتقدمة واستخدامها على سبيل المثال نموذج بسيط من حقيبة الكلمات Bag Of Words لمقارنة ترددات الكلمات / k-mers k-mers / frequencies of words بين التسلسلات الموروثة من إنسان نياندرتال وتسلسل السلالة القديمة المنضبة depleted archaic ancestry.

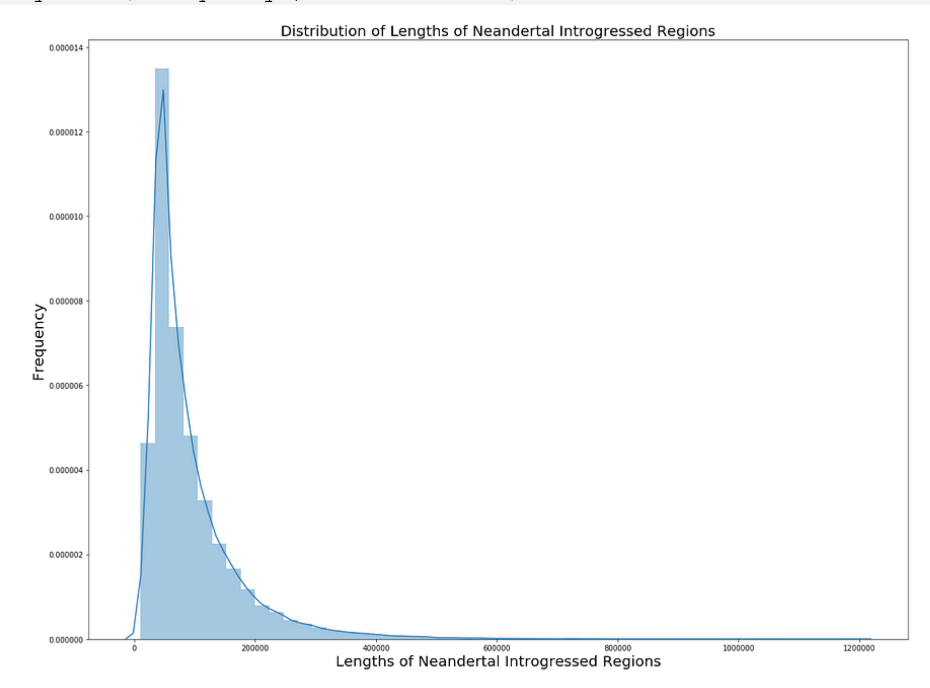
تحضير التسلسلات لتحليل المشاعر

الآن دعونا نوضح كيفية استخدام التعلم الآلة/العميق ومعالجة اللغة الطبيعية (NLP) عمليًا لتحديد مناطق إدخال الإنسان البدائي في الجينوم البشري الحديث. سأقوم هنا بصياغة المشكلة التالية لحلها:

أرني جزءًا من الحمض النووي الخاص بك وسوف أتوقع مدى احتمالية توريثه من إنسان نياندرتال

كمجموعة بيانات تدريبية، سنستخدم إحداثيات المناطق المرشحة لإدخال الإنسان البدائي Neanderthal introgression من Vernot و Akey ، علم 2016 تم تحديده باستخدام – S * statistic على الأوروبيين والآسيويين من مشروع 1000 Genomes، يمكن تنزيل البيانات من هنا. استخدمنا ملف أنماط الفرد المتقدم introgressed haplotypes file LL.callsetEUR.mr_0.99.neand_calls_by_hap.bed.merged.by_chr.bed واخترنا إحداثيات فريدة فقط، لذلك انتهى بنا المطاف مع 83601 منطقة من أصل إنسان نياندرتالفي الأوروبيين المعاصرين. دعونا نقرأ الإحداثيات ونلقي نظرة على توزيع الطول لمناطق تقدم الإنسان البدائي.

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
intr_coords = pd.read_csv('Akey_intr_coords.bed', header = None,
sep = "\t")
intr_lengths = intr_coords.iloc[:, 2]-intr_coords.iloc[:, 1]
sns.distplot(intr_lengths)
plt.title("Distribution of Lengths of Neandertal Introgressed
Regions", fontsize = 20)
plt.xlabel("Lengths of Neandertal Introgressed Regions", fontsize =
20)
plt.ylabel("Frequency", fontsize = 20)
```



يمكننا أن نرى أن طول مقاطع إدخال الإنسان البدائي يختلف من 10 كيلو بايت إلى 1.2 ميجا بايت في الثانية بمتوسط ~ 100 كيلو بايت. سنقوم الآن باستخدام هذه الإحداثيات واستخراج التسلسلات الفعلية من نسخة hg19 من الجينوم المرجعي البشري الذي له تنسيق ملف fasta ويمكن تنزيله من هنا. يمكننا بالطبع استخدام Python لاستخراج التسلسل ولكن من الأسرع القيام بذلك باستخدام samtools المستندة إلى C++.

```
import subprocess
a = 0
with open('hg19_intr_regions.fa', 'a') as fp:
    for i in range(intr_coords.shape[0]):
        coord = str(str(intr_coords.iloc[i, 0]) + ':'
                    + str(intr_coords.iloc[i, 1]) + '-' +
str(intr_coords.iloc[i, 2]))
        subprocess.run(['samtools', 'faidx', 'hg19.fa.gz',
str(coord)], stdout = fp)
        a = a + 1
        if a%10000 == 0:
            print('Finished ' + str(a) + ' Neanderthal introgressed
haplotypes')
```

سنقوم الآن ببناء اطار بيانات Pandas بإحداثيات أجزاء خارج إحداثيات إدخال الإنسان البدائي. لهذا الغرض، سنقوم بشكل عشوائي برسم امتدادات الحمض النووي DNA stretches بنفس طول مناطق الإدخال على نفس الكروموسوم والتحقق مما إذا كانت هذه الامتدادات تتقاطع مع أي من المناطق المتقدمة introgressed regions. بهذه الطريقة، نقوم ببناء مجموعتين من الإحداثيات غير المتداخلة: المناطق المتقدمة introgressed والمستنفدة depleted.

```
import numpy as np
chr_sizes = pd.read_csv("hg19.fa.gz.fai", header = None, sep =
"\t")
chr_sizes = chr_sizes.drop([2, 3, 4], axis = 1)
chr_list = []; start_list = []; end_list = []
intr_lengths = list(intr_coords.iloc[:, 2] - intr_coords.iloc[:,
1])
a = 0
for i in range(intr_coords.shape[0]):
    chr_df =
intr_coords[intr_coords[0].isin([intr_coords.iloc[i,0]])]
    overlap = True
    while overlap == True:
        reg_start = np.random.randint(1, int(chr_sizes[chr_sizes[0]
==

intr_coords.iloc[i,0]].iloc[:,1]))
        reg_end = reg_start + intr_lengths[i]
        for j in range(chr_df.shape[0]):
            b1 = chr_df.iloc[j,1]
            b2 = chr_df.iloc[j,2]
            if (reg_start > b1 and reg_start < b2) or (reg_end > b1
and reg_end < b2) or \
```

```
            (b1 > reg_start and b1 < reg_end) or (b2 > reg_start
and b2 < reg_end):
                overlap = True
                break
            else:
                overlap = False
    chr_list.append(intr_coords.iloc[i,0])
    start_list.append(reg_start)
    end_list.append(reg_end)
    a = a + 1
    if a%10000 == 0:
            print('Finished ' + str(a) + ' Neanderthal introgressed
haplotypes')
depl_coords = pd.DataFrame({'0': chr_list, '1': start_list, '2':
end_list})
depl_coords.to_csv("Akey_depl_coords.bed", index = False, header =
False, sep = "\t")
```

بمجرد بناء إطار البيانات الذي يحتوي على مناطق من أصل إنسان نياندرتال المنضب، يمكننا مرة أخرى استخراج تسلسل الحمض النووي الفعلي من hg19 fasta المطابق للأجزاء المستنفدة باستخدام samtools، هنا أتخطى هذه الخطوة للإيجاز ولكن تحقق من نوتبوك Jupyter الكامل على github لرؤية التفاصيل. بإلقاء نظرة فاحصة على التسلسلات المتقدمة introgressed المستخرجة والمستنفدة depleted المستخرجة، يمكننا أن نلاحظ عددًا قليلاً من التسلسلات المحتوية على النوكليوتيدات. هذه هي البيانات المفقودة missing data في علم الجينوم، أي المواقف التي لم يتم حلها بواسطة تقنيات التسلسل. من أجل عدم السماح للبيانات المفقودة بالتأثير على تحليلنا، فقد حذفت التسلسلات التي تحتوي على نوكليوتيد N واحد على الأقل، ويمكن القيام بذلك بكفاءة في BioPython:

```
from Bio import SeqIO
intr_file = 'hg19_intr_regions.fa'
depl_file = 'hg19_depl_regions.fa'
a = 0; i = 0
with open('hg19_intr_clean.fa', 'a') as
intr_out,open('hg19_depl_clean.fa', 'a') as depl_out:
    for intr, depl in zip(SeqIO.parse(intr_file, 'fasta'),
SeqIO.parse(depl_file, 'fasta')):
        upper_intr = intr.seq.upper()
        upper_depl = depl.seq.upper()
        a = a + 1
        if a%10000 == 0:
            print('Finished ' + str(a) + ' entries')
        if 'N' not in str(upper_intr) and 'N' not in
str(upper_depl):
            intr.seq = upper_intr
            SeqIO.write(intr, intr_out, 'fasta')
            depl.seq = upper_depl
            SeqIO.write(depl, depl_out, 'fasta')
            i = i + 1
        else:
            continue
```

```
print('We have processed ' + str(a) + ' entries and written ' +
str(i)
      + ' entries to two fasta-files')
```

الآن تم إعداد البيانات لإدخالهافي تحليل المعالجة اللغوية الطبيعية. دعنا ننتقل إلى نموذج حقيبة الكلمات Bag Of Words البسيط، أي النظرفي الاختلافي تكرار k-mers بين تسلسلات Neanderthal المقدمة مقابل المستنفدة.

تحليل المشاعر: متقدم مقابل مستنفد

سنبدأ بتنسيق التسلسلات من ملفي fasta كنصوص مع k-mers محددة المسافات ككلمات. بسبب قيود ذاكرة الكمبيوتر المحمول، قرأت فقط 10000 نيوكليوتيد أولًا من كل تسلسل، ثم استخدمت دالة getKmers التي تقسم كل تسلسل إلى k-mers ودمجت k-mers بطريقة محددة بمسافة، لذلك في النهاية، كان لدي قائمة الجمل، كل منها يمثل قائمة الكلمات / k-mers.

```
from Bio import SeqIO
from Bio.Seq import Seq
intr_file = 'hg19_intr_clean.fa'; depl_file = 'hg19_depl_clean.fa';
e = 0
intr_seqs = []; depl_seqs = []
for intr, depl in zip(SeqIO.parse(intr_file, 'fasta'),
SeqIO.parse(depl_file, 'fasta')):
    cutoff = 10000
    my_intr_seq = str(intr.seq)[0:cutoff]
    my_depl_seq = str(depl.seq)[0:cutoff]
    intr_seqs.append(my_intr_seq)
    depl_seqs.append(my_depl_seq)
    e = e + 1
    if e%20000 == 0:
        print('Finished ' + str(e) + ' entries')

def getKmers(sequence, size):
    return [sequence[x:x+size].upper() for x in range(len(sequence)
- size + 1)]

kmer = 5
intr_texts = [' '.join(getKmers(i, kmer)) for i in intr_seqs]
depl_texts = [' '.join(getKmers(i, kmer)) for i in depl_seqs]
```

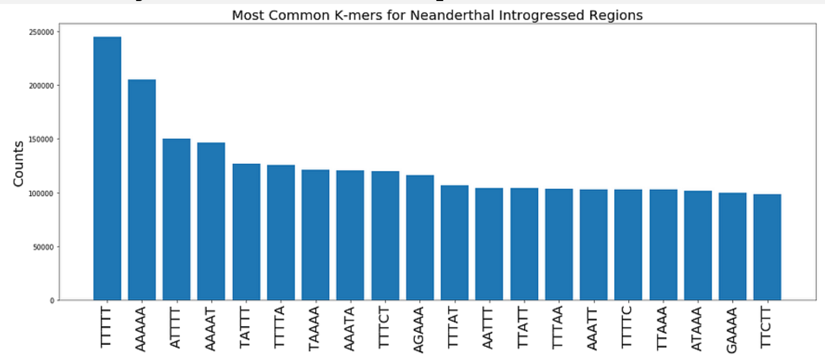
الآن يمكننا أن نتخيل بسهولة ترددات k-merفي متواليات Neanderthal المتقدمة والمستنفدة باستخدام فئة Counterفي Python من أجل عد الكلمات بكفاءة.

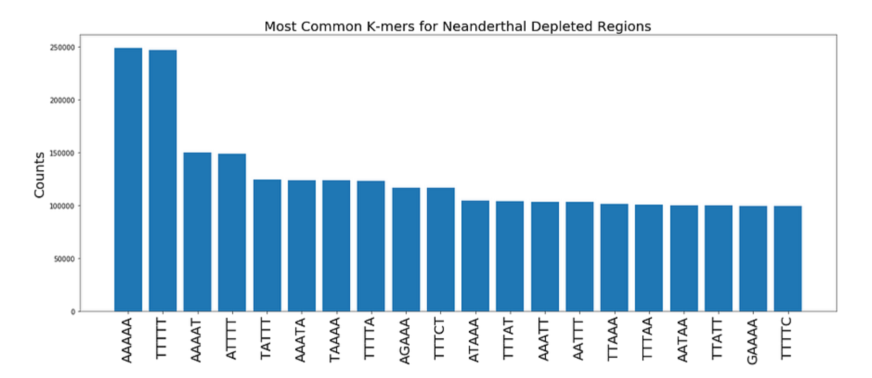
```
from collections import Counter
import matplotlib.pyplot as plt
fig = plt.figure(figsize = (20,18))
fig.subplots_adjust(hspace = 0.4, wspace = 0.4)

plt.subplot(2, 1, 1)
intr_sentences = [item.split(' ') for item in intr_texts]
D = dict(Counter([item for sublist in intr_sentences for item in
sublist]).most_common(20))
```

```
plt.bar(range(len(D)), list(D.values()), align='center')
plt.title('Most Common K-mers for Neanderthal Introgressed
Regions', fontsize = 20)
plt.ylabel("Counts", fontsize = 20); plt.xticks(rotation = 90)
plt.xticks(range(len(D)), list(D.keys()), fontsize = 20)

plt.subplot(2, 1, 2)
depl_sentences = [item.split(' ') for item in depl_texts]
D = dict(Counter([item for sublist in depl_sentences for item in
sublist]).most_common(20))
plt.bar(range(len(D)), list(D.values()), align='center')
plt.title('Most Common K-mers for Neanderthal Depleted Regions',
fontsize = 20)
plt.ylabel("Counts", fontsize = 20); plt.xticks(rotation = 90)
plt.xticks(range(len(D)), list(D.keys()), fontsize = 20)
```





على الرغم من أن -A و T-rich k-mers يبدو أنهما أكثر شيوعًا لكل من مناطق النياندرتال المتقدمة والمستنفدة، إلا أننا نلاحظ اختلافات صغيرة في تعداد k-mer بين الحالتين والتي يمكن أن تكون مؤشرًا لتكوين k-mer غير المتطابق في التسلسلات المتقدمة والمستنفدة. بعد ذلك، نقوم بترميز الكلمات / k-mers كأعداد صحيحة مع فئة CountVectorizer التي تبني ببساطة مفردات النص (عدد k-mers الفريدة) وتحسب تكرارات كل كلمة / k-mer في كل جملة / تسلسل.

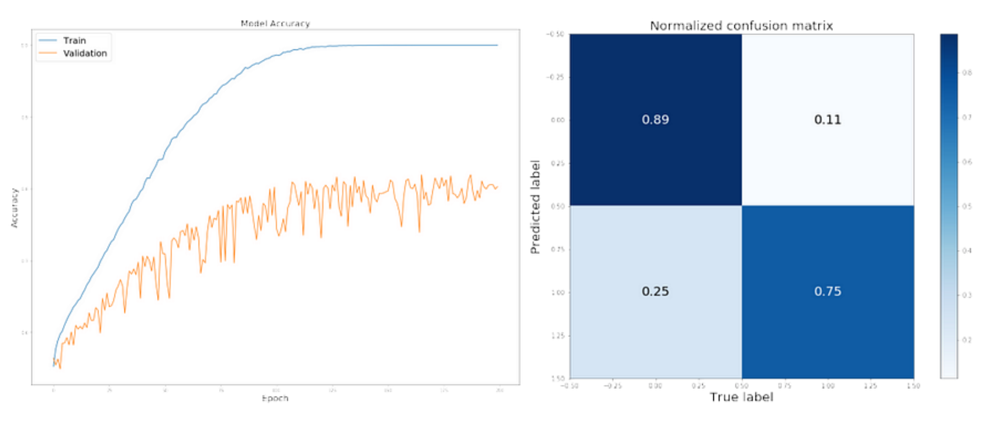
```
import numpy as np
from sklearn.model_selection import train_test_split
```

```
from sklearn.feature_extraction.text import CountVectorizer
merge_texts = intr_texts + depl_texts
labels = list(np.ones(len(intr_texts))) +
list(np.zeros(len(depl_texts)))
cv = CountVectorizer()
X = cv.fit_transform(merge_texts)
X = np.int32(X.toarray())
X_train, X_test, y_train, y_test = train_test_split(X, labels,
                                                     test_size =
0.20, random_state = 42)
```

بعد أن قمنا بتقسيم مجموعة البيانات إلى مجموعات فرعية للتدريب والاختبار، نحدد شبكة عصبية امامية التغذية feed-forward neural network باستخدام مُحسِّن Stochastic Gradient Descent (SGD) + Momentum و L1 الوزن المنظم (L1 weight regularizer).

```
from keras.models import Sequential
from keras.regularizers import l2, l1
from keras.callbacks import ModelCheckpoint
from keras.optimizers import SGD, Adam, Adadelta
from keras.layers import Dense, Flatten, Dropout

model = Sequential()
model.add(Dense(3000, input_shape = (X.shape[1], ), activation =
'sigmoid',
                kernel_regularizer = l1(0.00001)))
model.add(Dense(1, activation = 'sigmoid'))
sgd = SGD(lr = 0.0001, momentum = 0.9, nesterov = False)
model.compile(loss = 'binary_crossentropy', optimizer = sgd,
metrics = ['binary_accuracy'])
checkpoint = ModelCheckpoint("weights.best.hdf5",
monitor='val_binary_accuracy', verbose=1,
                             save_best_only = True, mode = 'max')
history = model.fit(X_train, y_train,
                    epochs = 200, verbose = 1, validation_split =
0.2, batch_size = 32,
                    shuffle = True, callbacks = [checkpoint])
```



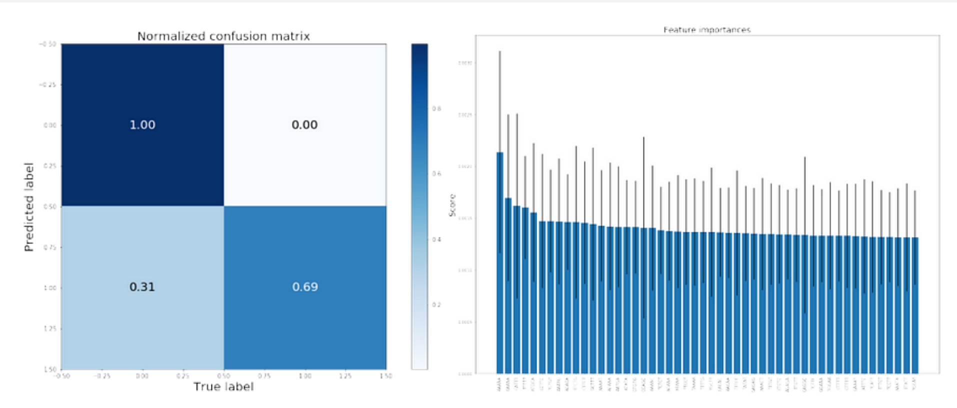
منحنى تدريب دقة MLP (يسار) ومصفوفة الارتباك للتقييم على مجموعة بيانات الاختبار (يمين)

وصل النموذج إلى دقة تصل إلى 82.2٪ على مجموعة بيانات الاختبار التي صنفت تسلسل النياندرتال المتقدم مقابل الأصل المستنفد. كانت هذه نتيجة أفضل بشكل واضح مقارنة بالنماذج الخطية linear models مثل الانحدار اللوجستي Logistic Regression أو متجهات الآلات الداعمة Support Vector Machines (SVM) أو مصنّف نايف بايز Naive Bayes Classifier، ومع ذلك، وبشكل غير متوقع، وصل مصنف الغابة العشوائية Random Forest إلى دقة أعلى بلغت 84.5٪ في نفس مجموعة بيانات الاختبار. عند عرض ميزات نموذج الغابة العشوائية، نلاحظ أن مثل k-mers مثل AAAAA وCAAAA وCATTT وTTTTT هو الأكثر تنبوًا. نحصل على حدس على الفور على أن التنبؤ بمناطق النياندرتال المتدخلة / المستنفدة له علاقة بمحتوى AT / GC لأن أكثر k-mers تنبؤية غنية للغاية بـ AT (AT-rich).

```
import pickle
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier

classifier = RandomForestClassifier(n_estimators = 500)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
pickle.dump(classifier, open('RF_model_Neand_Intr_vs_Depl.sav',
'wb'))

importances = classifier.feature_importances_
std = np.std([tree.feature_importances_ for tree in
classifier.estimators_], axis = 0)
indices = np.argsort(importances)[::-1]
plt.title("Feature importances", fontsize = 20)
plt.bar(range(X_train.shape[1])[0:50], importances[indices][0:50],
        yerr = std[indices][0:50], align = "center")
plt.xticks(rotation = 90); plt.ylabel('Score', fontsize = 20)
plt.xticks(range(X_train.shape[1])[0:50],
np.array(names)[indices][0:50])
```



مصفوفة الارتباك لمصنف الغابة العشوائية (يسار) وخصائص الاستيراد التي يهيمن عليها AT-rich k-mers (يمين)

من مصفوفة الارتباك لمصنف الغابة العشوائية (أعلاه)، يتضح أن النموذج كان يتمتع بدقة عالية جداً عند التنبؤ بأجزاء من أصل إنسان نياندرتال المستنفد، ومع ذلك، كان أداؤه ضعيفاً للغاية (أسوأ من الشبكة العصبية)في تصنيف المناطق المتقدمة introgressed regions.

توقع الجينات الموروثة من إنسان نياندرتال

الآن دعونا نعود إلى مسألة التنبؤ من تسلسل DNA معين سواء كان موروثًا من إنسان نياندرتال أم لا. سأقوم هنا بتوليد مثل هذه التنبؤات لجينات ترميز البروتين البشري. لنقم بتنزيل ملف شرح الجين RefSeq لـ hg19 هنا، قم بتنظيفه واستخدام إحداثيات الجينات لاستخراج تسلسلها من ملفات الجينوم المرجعي fasta بالمثل لمعرفة كيفية قيامنا بذلك للمناطق المتقدمة والمستنفدة، انظر التفاصيل في نوتبوك Jupyter على github. مرة أخرى نقوم ببناء نصوص الجينات عبر تسلسل محدد بفراغ لـ k-mers.

```
from Bio import SeqIO
gene_seqs = []; gene_ids = [], a = 0
for gene in SeqIO.parse('hg19_gene_clean.fa', 'fasta'):
    cut = 8800
    if len(str(gene.seq)) < cut:
        continue
    s_gene = str(gene.seq)[0:cut]
    if s_gene.count('A')>0 and s_gene.count('C')>0 and
s_gene.count('G')>0 \
    and s_gene.count('T')>0:
        gene_seqs.append(s_gene)
        gene_ids.append(str(gene.id))
    a = a + 1
    if a%10000 == 0:
        print('Finished ' + str(a) + ' genes')

def getKmers(sequence, size):
    return [sequence[x:x+size].upper() for x in range(len(sequence)
- size + 1)]

kmer = 5
gene_texts = [' '.join(getKmers(i, kmer)) for i in gene_seqs]
```

بعد ذلك، سنستخدم مصنف Random Forest المدربين لتوليد تنبؤات لتسلسل الجينات التي تم تحويلها سابقًا إلى نصوص. وبالتالي، يتم تحويل نصوص الجينات إلى أعداد صحيحة باستخدام CountVectorizer.

```
import pickle
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer

classifier = pickle.load(open('RF_model_Neand_Intr_vs_Depl.sav',
'rb'))
cv = CountVectorizer(); X_gene = cv.fit_transform(gene_texts)
gene_predictions = classifier.predict(X_gene.toarray())
```

```
gene_predictions_prob = classifier.predict_proba(X_gene.toarray())
gene_predictions_prob_0 = [i[0] for i in gene_predictions_prob]
gene_predictions_prob_1 = [i[1] for i in gene_predictions_prob]

gene_ids = []; gene_symbol = []
with open('gene_ids.txt','r') as fin:
    for line in fin:
        line = line.split('\t')
        gene_ids.append(line[0])
        gene_symbol.append(line[1].rstrip())

gene_pred_df = pd.DataFrame({'Gene': gene_ids, 'Gene_Symbol':
gene_symbol,
                            'Predict': gene_predictions,
                            'Prob_0': gene_predictions_prob_0,
                            'Prob_1': gene_predictions_prob_1})
gene_pred_df = gene_pred_df.sort_values(['Prob_1'], ascending =
False)
```

	Gene	Gene_Symbol	Predict	Prob_0	Prob_1
12554	chr3:19988572-20026667	RAB5A	1.0	0.032	0.968
16186	chr6:961241-1101567	LINC01622	1.0	0.046	0.954
2966	chr11:4665156-4676716	OR51E1	1.0	0.066	0.934
15270	chr5:57878871-58155222	RAB3C	1.0	0.094	0.906
10923	chr2:179694484-179914786	CCDC141	1.0	0.094	0.906
6014	chr14:101355986-101465450	MEG8	1.0	0.100	0.900
16252	chr6:8435856-8785678	LOC100506207	1.0	0.136	0.864
2440	chr10:87359312-88126250	GRID1	1.0	0.136	0.864
16251	chr6:8435856-8712526	LOC100506207	1.0	0.136	0.864
382	chr1:33231235-33240571	KIAA1522	1.0	0.154	0.846
12301	chr22:32870707-32894818	FBXO7	1.0	0.158	0.842
19128	chr8:17154306-17271040	MTMR7	1.0	0.168	0.832
4853	chr12:96043031-96067770	PGAM1P5	1.0	0.168	0.832
12597	chr3:29322803-30032809	RBMS3	1.0	0.174	0.826
12598	chr3:29322803-30051886	RBMS3	1.0	0.174	0.826
16047	chr5:169064251-169510386	DOCK2	1.0	0.176	0.824
13843	chr3:191857182-192445388	FGF12	1.0	0.178	0.822
13842	chr3:191857182-192126838	FGF12	1.0	0.178	0.822
16878	chr6:80340822-80413387	SH3BGRL2	1.0	0.180	0.820
3739	chr11:95523625-95565857	CEP57	1.0	0.182	0.818
10325	chr2:86730553-86790620	CHMP3	1.0	0.186	0.814
10326	chr2:86730553-86948245	RNF103-CHMP3	1.0	0.186	0.814
2238	chr10:61786056-61900774	ANK3	1.0	0.192	0.808
2239	chr10:61786056-62149742	ANK3	1.0	0.192	0.808
2240	chr10:61786056-62332714	ANK3	1.0	0.192	0.808
2241	chr10:61786056-62493284	ANK3	1.0	0.192	0.808
3761	chr11:102391239-102401484	MMP7	1.0	0.194	0.806
13506	chr3:147795946-147805816	LINC02032	1.0	0.200	0.800
10404	chr2:99235569-99279936	MGAT4A	1.0	0.214	0.786
10405	chr2:99235569-99347589	MGAT4A	1.0	0.214	0.786

	Gene	Gene_Symbol	Predict	Prob_0	Prob_1
9274	chr19:42901280-42912604	LOC101930071	0.0	0.876	0.124
18	chr1:1413495-1431584	ATAD3B	0.0	0.876	0.124
18318	chr7:66147078-66276448	RABGEF1	0.0	0.876	0.124
9412	chr19:48958964-48969367	KCNJ14	0.0	0.878	0.122
12785	chr3:47537130-47555199	ELP6	0.0	0.880	0.120
4110	chr12:6420099-6437672	PLEKHG6	0.0	0.880	0.120
1149	chr1:155204239-155214653	GBA	0.0	0.884	0.116
1744	chr1:235330210-235491532	ARID4B	0.0	0.884	0.116
1743	chr1:235330210-235490802	ARID4B	0.0	0.884	0.116
13	chr1:1288069-1298921	MXRA8	0.0	0.884	0.116
3471	chr11:64948686-64979477	CAPN1	0.0	0.886	0.114
5137	chr12:123259056-123311927	CCDC62	0.0	0.886	0.114
17953	chr7:5965777-6010314	RSPH10B	0.0	0.888	0.112
19927	chr8:145597704-145618453	ADCK5	0.0	0.888	0.112
6054	chr14:105956192-105965585	C14orf80	0.0	0.888	0.112
17952	chr7:5965777-6010314	RSPH10B	0.0	0.888	0.112
8047	chr17:73720776-73753899	ITGB4	0.0	0.890	0.110
19920	chr8:145106167-145115606	OPLAH	0.0	0.892	0.108
11708	chr20:32319566-32380075	ZNF341	0.0	0.894	0.106
8634	chr19:496490-505343	MADCAM1	0.0	0.898	0.102
1759	chr1:236558716-236648008	EDARADD	0.0	0.904	0.096
20487	chr9:131857073-131873077	CRAT	0.0	0.906	0.094
6935	chr16:66638228-66647795	CMTM3	0.0	0.906	0.094
17725	chr6_mcf_hap5:3144866-3154459	LSM2	0.0	0.908	0.092
17803	chr6_qbl_hap6:3058814-3068398	LSM2	0.0	0.908	0.092
17608	chr6_dbb_hap3:3050751-3060344	LSM2	0.0	0.910	0.090
6055	chr14:105956520-105965585	C14orf80	0.0	0.910	0.090
17513	chr6_cox_hap2:3274737-3284332	LSM2	0.0	0.910	0.090
16482	chr6:31765169-31774761	LSM2	0.0	0.910	0.090
19906	chr8:144873090-144897549	SCRIB	0.0	0.916	0.084

يُتوقع أن يكون للجينات احتمالية عالية (يسار) ومنخفضة (يمين) لتورثها من إنسان نياندرتال

بهذه الطريقة، ننتج لكل جين احتمال (Prob_1) أن يتم توريثه من إنسان نياندرتال وأن يكون جينًا بشريًا أصليًا (Prob_0). يمكننا بسهولة عرض الجينات ذات أعلى Prob_1 (أدنى Prob_0) وأعلى Prob_0 (أدنى Prob_1)، يرجى الاطلاع على قوائم الجينات أعلاه. بشكل ملحوظ، من بين 22000

الجينية المشفرة للبروتين فقط، كان من المتوقع أن يحتوي 477 جينة فقط على الحمض النووي لإنسان نياندرتال. هذه ملاحظة شيقة للغاية، وسوف نعيد النظر فيها لاحقًا.

تصور K-mer / Word Embeddings

يمكن تصور مفردات النصوص الجينومية التي تم إنتاجها من خلال ربط k-mers عبر نموذج Word Word2Vec Embedding الذي يقدم تخمينات ذكية حول التشابه بين الكلمات (k-mers في حالتنا). بمجرد ملاءمته لبياناتنا، يمثل Word2Vec كل k-mer كمتجه كامن من 100 بعد، يمكن النظر إلى المتجهات على أنها تمثيل رقمي آخر لبياناتنا التي يمكن إدخالها في أي تقنية لتقليل الأبعاد dimension reduction لمزيد من الاستكشاف. سنقوم هنا بتضمين وتصور باستخدام UMAP وtSNE القيم k المستخدمة في تحليل مشاعر مقدمة الإنسان البدائي مقابل الاستنفاد. يمكننا اكتشاف مجموعتين واضحتين من مجموعات k-mer. تُظهر نظرة فاحصة أن أحدهما يبدو أنه غني بالـ k-mers الغني بـ AT والآخر (الأصغر) يتكون في الغالب من k-mers الغنية بـ GC.

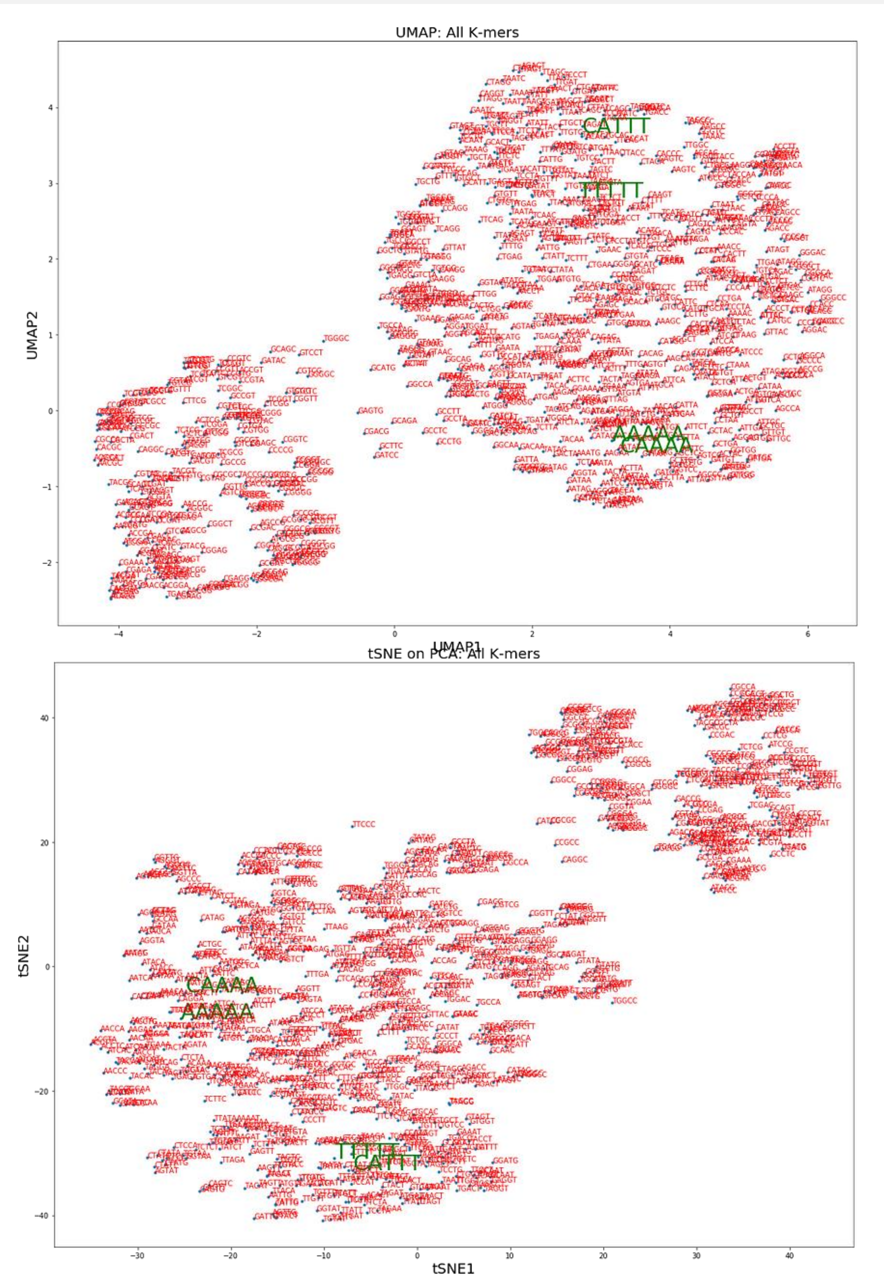
```
from umap import UMAP
import matplotlib.pyplot as plt
from gensim.models import Word2Vec

kmer = 5
sequences = intr_seqs + depl_seqs; sentences = []
for i in range(len(sequences)):
    sentences.append(getKmers(sequences[i], kmer))

model = Word2Vec(sentences, min_count = 2, workers = 4)
X = model[model.wv.vocab]

X_reduced = PCA(n_components = 5).fit_transform(X)
umap_model = UMAP(n_neighbors = 30, min_dist = 0.2, n_components =
2)
umap = umap_model.fit_transform(X_reduced)
plt.scatter(umap[:, 0], umap[:, 1], s = 10, cmap = 'tab10')
plt.title('UMAP: All K-mers', fontsize = 20)
plt.xlabel("UMAP1", fontsize = 20); plt.ylabel("UMAP2", fontsize =
20)
words = list(model.wv.vocab)
for i, word in enumerate(words):
    if word == 'AAAAA':
        plt.text(umap[i, 0], umap[i, 1], word, fontsize = 30, c =
'green')
    elif word == 'CAAAA':
        plt.text(umap[i, 0], umap[i, 1], word, fontsize = 30, c =
'green')
    elif word == 'CATTT':
        plt.text(umap[i, 0], umap[i, 1], word, fontsize = 30, c =
'green')
    elif word == 'TTTTT':
        plt.text(umap[i, 0], umap[i, 1], word, fontsize = 30, c =
'green')
```

```
    else:
        plt.text(umap[i, 0], umap[i, 1], word, fontsize = 10, c =
'red')
```

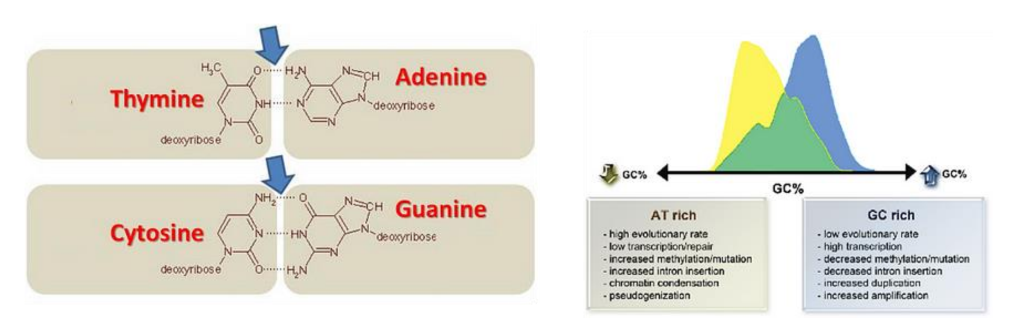


لقد سلطت الضوء على وجه التحديد من خلال اللون الأخضر على أكثر k-mers تنبؤية وفقًا لمصنف Random Forest الذي يميز بين تسلسلات Neanderthal المقدمة مقابل المستنفدة. كما هو

متوقع، يقعون جميعًافي مجموعة AT-rich الأكبر. يعد تصور تضمين Word هذا دليلًا آخر على وجود بعض الهياكل الموجودةفي جمل k-mer، والتي يمكنها التنبؤ بأصل الإنسان البدائي ولها علاقة بالتوازن بين أجزاء الحمض النووي الغنية بالـ GC وAT.

التطور يزيل الحمض النووي لإنسان نياندرتال من الجينات

بتلخيص كل شيء تمت مناقشتهفي الأقسام السابقة، يمكن للمرء أن يستنتج أن وجود أصل إنسان نياندرتالفي حمضنا النووي يرتبط بشكل مثير للريبة بمحتوى AT / GC عبر الجينوم. من المعروف أن جيناتنا غنية بالـ GC، ومحتوى GC حوالي 47٪ مقارنة بـ 41٪ على مستوى الجينوم، وهو اختلاف كبيرفي محتوى GC.



عادة ما تكون الجينات البشرية غنية بالـ GC وهو ما يدل على عدم وجود سلالة نياندرتال

لذلك يمكننا أن نسأل عن مدى تداخل مناطق التقديم introgression واستنفاد depletion الإنسان البدائي مع الجينات؟ يمكننا حساب عدد التقاطعات بين الجينات وإحداثيات التداخل introgression coordinates مع bedtools intersect. ومع ذلك، نظرًا لأن إحداثيات المناطق المستنفدة للإنسان البدائي تم اختيارها مسبقًا عشوائيًا (طلبنا فقط عدم تداخلها مع إحداثيات التقديم introgression coordinates)، يجب أن نكرر إجراء الاختيار عدة مرات لتقدير أهمية التقاطعات.

```
import subprocess
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

perm_n = []
for k in range(20):
    chr_list = []; start_list = []; end_list = []
    intr_lengths = list(intr_coords.iloc[:, 2] -
intr_coords.iloc[:, 1])
    a = 0
    for i in range(intr_coords.shape[0]):
        chr_df =
intr_coords[intr_coords[0].isin([intr_coords.iloc[i,0]])]
        overlap = True
        while overlap == True:
```

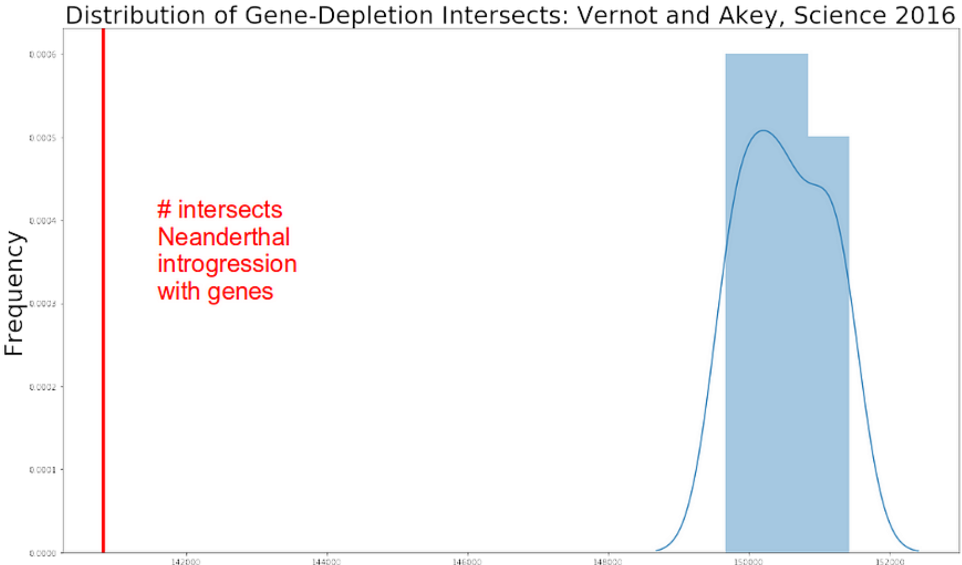
```
            reg_start=

np.random.randint(1,int(chr_sizes[chr_sizes[0]==intr_coords.iloc[i,
0]].iloc[:,1]))
            reg_end = reg_start + intr_lengths[i]
            for j in range(chr_df.shape[0]):
                b1 = chr_df.iloc[j,1]; b2 = chr_df.iloc[j,2]
                if (reg_start > b1 and reg_start < b2) or (reg_end
> b1 and reg_end < b2) \
                or (b1 > reg_start and b1 < reg_end) or (b2 >
reg_start and b2 < reg_end):
                    overlap = True
                    break
                else:
                    overlap = False
        chr_list.append(intr_coords.iloc[i,0])
        start_list.append(reg_start); end_list.append(reg_end)
        a = a + 1
        if a%20000 == 0:
            print('Finished ' + str(a) + ' Neanderthal haplotypes')
    depl_coords = pd.DataFrame({'0': chr_list, '1': start_list,
'2': end_list})
    depl_coords.to_csv("depl_temp.txt", index = False, header =
False, sep = "\t")

    with open('n_intersects.txt', 'w') as fp:
        subprocess.run(['bedtools', 'intersect', '-a',
'depl_temp.txt', '-b',
                        'gene_coords.txt'], stdout = fp)
    akey_n = pd.read_csv('n_intersects.txt', header = None, sep =
"\t")
    print(k, akey_n.shape[0])

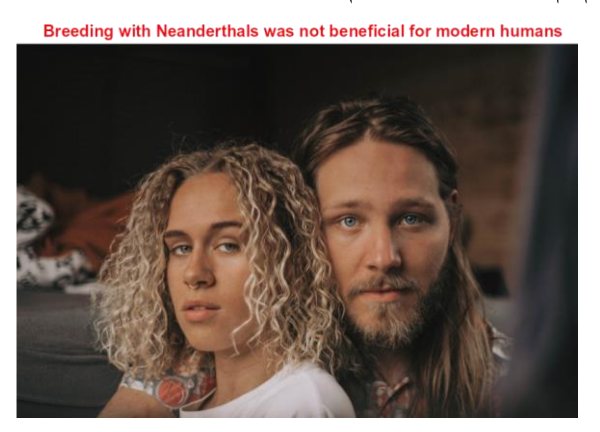
print('**********************************************************')
    perm_n.append(akey_n.shape[0])

plt.axvline(x = 140821, linewidth = 4, color = 'r')
sns.distplot(perm_n)
plt.title("Distribution of Gene-Depletion Intersects: Vernot and
Akey, Science 2016",
          fontsize = 30)
plt.xlabel("Number of Intersects Between Gene and Neanderthal
Depleted Regions",
           fontsize = 30)
plt.ylabel("Frequency", fontsize = 30)
```



يمكننا أن نرى أن عدد التقاطعات بين الجينات ومقاطع تقديم الإنسان البدائي Neanderthal introgression segments، 140821، أقل بكثير من عدد التقاطعات بين الجينات ومناطق النياندرتال المستنفدة Neanderthal depleted regions.في الواقع، لم يكن أي من الـ 17 منطقة المرسومة عشوائيًا لاستنفاد إنسان نياندرتال بها عدد من التداخلات مع الجينات أقل من 140821 أو يساوي 140821، لذلك يمكننا حساب القيمة p كقيمة 17 / 1> p. ما نراه هو أن الإنسان البدائي تقع المناطق المُدخلةفي الغالب خارج الجينات، مما يعني أن التطور لم يعطي الأولوية لسلالة الإنسان البدائي وحاول إبعاده عن العناصر الأكثر وظيفيةفي جينومنا، وهي الجينات المشفرة للبروتين. لذلك، ولأسباب غامضة، فإن التهجين مع إنسان نياندرتال لم يحسن لياقتنا البدنية وحاول التطور تصحيح ذلك. يتحدثون كثيرًافي الصحف والمجلات عن حقيقة رائعة جدًا تتعلق بالتزاوج بين البشر المعاصرين والنياندرتال، لكنهم لم يذكروا أبدًا أن هذا لم يكن مفيدًا للإنسان الحديث، أليس كذلك؟

Breeding with Neanderthals was not beneficial for modern humans



الاستنتاج

في هذا المنشور، تعلمنا عن الإمكانات الهائلة للتعلم الآلي / العميق ومعالجة اللغة الطبيعية (NLP) لبيولوجيا التطور ومجالات أبحاث الحمض النووي القديمة، والتي لا تزال غير مستخدمة بالكامل. توفر بيانات الجينوم أحد المصادر الرئيسية للمعلوماتفي علم الأحياء التطوري وتتألف من ملايين ومليارات من قطع الحمض النووي القصيرة التي يمكن وينبغي تحليلها باستخدام التعلم العميق. أوضحنا هنا كيفية إعداد بيانات الجينوم لـ NLP وتحليلها باستخدام حقيبة الكلمات Bag Of Words وتضمين الكلمة Word Embedding . قمنا بتدريب نموذج قادر على التنبؤ بما إذاكان التسلسل الجيني موروثًا من إنسان نياندرتال. باستخدام النموذج، قمنا ببناء قوائم بالجينات التي من المحتمل أن تكون موروثة من إنسان نياندرتال واكتشفنا عدم وجود سلالة قديمةفي جيناتنا مما يشير إلى أن التزاوج مع إنسان نياندرتال كان تطوريًا، ولم يكن مفيداً للإنسان الحديث.

المصدر:

https://towardsdatascience.com/deep-learning-on-neanderthal-genes-ad1478cf37e7

14) LSTM لاكتشاف DNA لإنسان نياندرتال LSTM to Detect Neanderthal DNA

(الشبكات العصبية طويلة الذاكرة لعلم الجينوم القديم)

هذا هو المنشور الثامن من عمودي التعلم العميق لعلوم الحياة حيث أوضح كيفية استخدام التعلم العميق للحمض النووي القديم، وبيولوجيا الخلية المفردة، وتكامل بيانات OMIC، والتشخيص السريري والتصوير المجهري. فيما بعد التعلم العميق حول جينات الإنسان البدائي Neanderthal Genes، أكدت على الإمكانات الهائلة للتعلم العميق و المعالجة اللغوية الطبيعية لعلم الجينوم القديم وأوضحت كيفية البدء عمليًافي استخدامه لاستنتاج مناطق من إدخال الإنسان البدائيفي الجينوم البشري الحديث. سنقوم الآن بتطبيق القوة الكاملة للشبكات العصبية المتكررة Recurrent Neural Networks (RNNs) ذات الذاكرة الطويلة من أجل تنبؤ أفضل لشرائح من أصل إنسان نياندرتالفي الجينوم البشري الحديث.

HMM مقابل LSTM لعلم الجينوم القديم

الحمض النووي DNA عبارة عن تسلسل ذي ذاكرة طويلة تتجلى من خلال الارتباطات طويلة المدى على طول التسلسل الذي يُسمى اختلال التوازن الارتباطي Linkage Disequilibrium. ومع ذلك، يتم إجراء الكثير من التحليلاتفي علم الجينوم القديم باستخدام نموذج ماركوف الخفي Hidden Markov Model (HMM) وهو نموذج بلا ذاكرة لا يمكن أن يأخذ سوى بضع خطوات سابقةفي الاعتبار. لقد ثبت أن الشبكات العصبية الاصطناعية للذاكرة طويلة قصيرة المدى Long-Short Term Memory (LSTM) تتفوق على HMMفي عدد من التطبيقات مثل التعرف على الكلام speech recognition وتوليد النص text generation وما إلى ذلك، من خلال الاستفادة من قدرتها الفريدة على حفظ العديد من الخطوات السابقةفي تسلسل يوفر ذلك ما يكفي البيانات المتاحة.



يمثل علم الجينوم Genomics وعلم الجينوم القديم Ancient Genomics مصدرًا حقيقيًا للبيانات الضخمة Big Data بفضل تسلسل الجيل التالي Next Generation Sequencing (NGS) الذي يوفر ملايين ومليارات من التسلسلات التي يمكن استخدامها كأمثلة تدريبية / ملاحظات إحصائية لتدريب LSTMs. لذلك فإن بيانات الجينوم القديم هي هدية للتعلم العميق!

على الرغم من ازدهار موارد التدريب في علم الجينوم / علم الجينوم القديم، لا يزال يتم إجراء غالبية التحليلات على بيانات الجينوم المحاكية simulated genomic data، وربما تكون نظرية الاندماج coalescent theory هي الإطار الأكثر شيوعًا لمحاكاة الديموغرافيا والاختيار في علم الوراثة السكانية Population Genetics.

LSTM + تضمين الكلمة على الحمض النووي للإنسان البدائي

سأقوم هنا بتوسيع الأفكار حول استخدام المعالجة اللغوية الطبيعية NLP للجينوميات القديمة التي تم التعبير عنها في رسالتي السابقة التعلم العميق على جينات الإنسان البدائي وإثبات تفوق LSTMs في تحليل بيانات التسلسل. كما لاحظت، فإن التعلم العميق من المنشور السابق لم يكن "عميقًا Deep " بشكل خاص بل "واسع Wide "، بالإضافة إلى ذلك، يبدو أن Random Forest تظهر أداءً أفضل في نموذج حقيبة الكلمات Bag of Words. سأقوم هنا بتنفيذ نموذج LSTM يصل إلى دقة تصل إلى 99٪ في الكشف عن امتدادات الحمض النووي الموروثة من إنسان نياندرتال. لقد أوضحت كيفية استخراج تسلسلات النياندرتال المتقدمة والمستنفدة في المنشور السابق، لذلك سأبدأ هنا بقراءة التسلسلات، وتقطيعها إلى 200 سلسلة فرعية طويلة من النيوكليوتيدات، كل منها يمثل جملة حتى نتمكن من تقسيم كل جملة إلى k-mers / كلمات الجمل.

```
from Bio import SeqIO
from Bio.Seq import Seq

intr_seqs = []; depl_seqs = []; e = 0
intr_f = 'hg19_intr_clean.fa'; depl_f = 'hg19_depl_clean.fa'
for intr, depl in zip(SeqIO.parse(intr_f, 'fasta'),
SeqIO.parse(depl_f, 'fasta')):

    step = 200; jump = 1; a = 0; b = step; n_jumps = 5
    for j in range(n_jumps):
        s_intr = str(intr.seq)[a:b]; s_depl = str(depl.seq)[a:b]
        intr_seqs.append(s_intr); depl_seqs.append(s_depl)
        a = a + jump; b = a + step

    e = e + 1
    if e%20000 == 0:
        print('Finished ' + str(e) + ' entries')

def getKmers(sequence, size):
    return [sequence[x:x+size].upper() for x in range(len(sequence)
- size + 1)]
```

```
kmer = 10
intr_texts = [' '.join(getKmers(i, kmer)) for i in intr_seqs]
depl_texts = [' '.join(getKmers(i, kmer)) for i in depl_seqs]
```

نتهي مع 737340 جملة تنتمي إلى فئتين: إنسان نياندرتال متقدم introgressed ومنضب depleted. الخطوة التالية هي ترميز الجمل مرة واحدة عن طريق تحويل k-mers / الكلمات إلى أعداد صحيحة باستخدام كلاس Tokenizer في Python. لاحظ أن الحشو padding ليس ضروريًا حقًا في حالتنا لأن جميع الجمل لها نفس الطول، بطول 191 ألف متر، لكني أدرجها هنا للتعميم.

```
import numpy as np
from keras.preprocessing.text import Tokenizer
from sklearn.model_selection import train_test_split
from keras.preprocessing.sequence import pad_sequences

merge_texts = intr_texts + depl_texts
labels = list(np.ones(len(intr_texts))) +
list(np.zeros(len(depl_texts)))

tokenizer = Tokenizer()
tokenizer.fit_on_texts(merge_texts)
encoded_docs = tokenizer.texts_to_sequences(merge_texts)
max_length = max([len(s.split()) for s in merge_texts])
X = pad_sequences(encoded_docs, maxlen = max_length, padding =
'post')

X_train,X_test,y_train,y_test = train_test_split(X,labels,

test_size=0.20,random_state=42)
vocab_size = len(tokenizer.word_index) + 1
```

كان حجم المفردات 964114 في حالتنا وهو أقل من 4 ^ 10 = 1 048 576 مما يعني أنه لا توجد كل المفردات العشرة الممكنة المكونة من 4 أحرف داخل التسلسلات. أخيرًا، نحدد نموذجًا تسلسليًا بدءًا من طبقة التضمين Embedding Layer من Keras الذي يتعلم تضمين الكلمة Word Embeddings أثناء تصنيف التسلسلات ، متبوعًا بـ ثنائي الاتجاه LSTM وطبقات كثيفة Dense layers.

```
from keras.models import Sequential
from keras.callbacks import ModelCheckpoint
from keras.optimizers import SGD, Adam, Adadelta, RMSprop
from keras.layers import Embedding, LSTM, Dense, Bidirectional

model = Sequential()
model.add(Embedding(vocab_size, 10))
model.add(Bidirectional(LSTM(10)))
model.add(Dense(10, activation = 'relu'))
model.add(Dense(1, activation = 'sigmoid'))

epochs = 5
model.compile(loss='binary_crossentropy',optimizer='rmsprop',metric
s=['accuracy'])
```

```
checkpoint=ModelCheckpoint("LSTM.weights.best.hdf5",
monitor='val_acc',verbose = 1,
                             save_best_only = True, mode = 'max')
model.summary()
```

```
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, None, 10)          9641140
_________________________________________________________________
bidirectional_2 (Bidirection (None, 20)                1680
_________________________________________________________________
dense_3 (Dense)              (None, 10)                210
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 11
=================================================================
Total params: 9,643,041
Trainable params: 9,643,041
Non-trainable params: 0
_________________________________________________________________
None
```

ميزة استخدام تضمين الكلمة Word Embeddings في الطبقة الأولى هي تقليل الأبعاد من 964114 إلى 10 أبعاد فقط. هذا يقلل من الضبط الزائد overfitting ويحسن قابلية تعميم generalizability النموذج عن طريق إجبار كلمات مماثلة على مشاركة المعلمات / الأوزان الملائمة. الآن يمكننا البدء في تدريب نموذجنا.

```
import matplotlib.pyplot as plt
history = model.fit(X_train, y_train,
                    epochs = epochs, verbose = 1, validation_split
= 0.2,
                    batch_size = 32, shuffle = True,
                    callbacks = [checkpoint])

plt.figure(figsize=(20,15))
plt.plot(history.history['loss']);
plt.plot(history.history['val_loss'])
plt.title('Model Loss', fontsize = 20)
plt.ylabel('Loss', fontsize = 20); plt.xlabel('Epoch', fontsize =
20)
plt.legend(['Train', 'Validation'], fontsize = 20)

plt.figure(figsize=(20,15))
plt.plot(history.history['acc']);
plt.plot(history.history['val_acc'])
plt.title('Model Accuracy', fontsize = 20)
plt.ylabel('Accuracy', fontsize = 20); plt.xlabel('Epoch', fontsize
= 20)
plt.legend(['Train', 'Validation'], fontsize = 20)
```

```
Train on 471897 samples, validate on 117975 samples
Epoch 1/5
471897/471897 [==============================] - 3035s 6ms/step - loss: 0.2911 - acc: 0.8573 - val_loss: 0.0965 - v
al_acc: 0.9636

Epoch 00001: val_acc improved from -inf to 0.96362, saving model to weights.best.hdf5
Epoch 2/5
471897/471897 [==============================] - 5225s 11ms/step - loss: 0.0371 - acc: 0.9869 - val_loss: 0.0850 -
val_acc: 0.9716

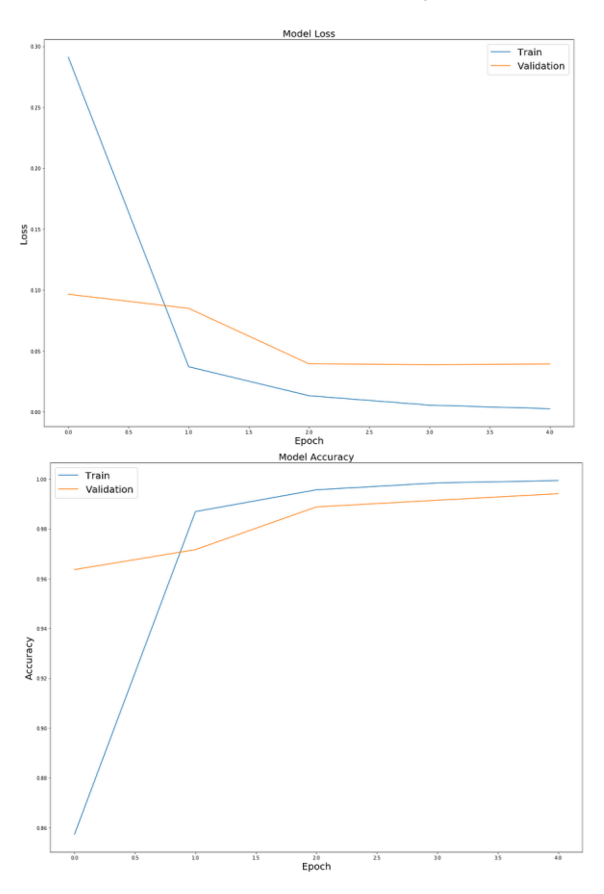
Epoch 00002: val_acc improved from 0.96362 to 0.97165, saving model to weights.best.hdf5
Epoch 3/5
471897/471897 [==============================] - 3116s 7ms/step - loss: 0.0133 - acc: 0.9957 - val_loss: 0.0394 - v
al_acc: 0.9888

Epoch 00003: val_acc improved from 0.97165 to 0.98881, saving model to weights.best.hdf5
Epoch 4/5
471897/471897 [==============================] - 3172s 7ms/step - loss: 0.0055 - acc: 0.9984 - val_loss: 0.0388 - v
al_acc: 0.9915

Epoch 00004: val_acc improved from 0.98881 to 0.99149, saving model to weights.best.hdf5
Epoch 5/5
471897/471897 [==============================] - 3160s 7ms/step - loss: 0.0026 - acc: 0.9994 - val_loss: 0.0393 - v
al_acc: 0.9941

Epoch 00005: val_acc improved from 0.99149 to 0.99411, saving model to weights.best.hdf5
```

كان كافيًا إجراء تدريب لمدة 5 فترات epochs فقط لأن النموذج سرعان ما وصل إلى دقة مذهلة بلغت 99٪ في مجموعة بيانات التحقق من الصحة validation data.



عند تقييم أداء النموذج على مجموعة بيانات الاختبار test data، حصلنا مرة أخرى على 99٪ من دقة تصنيف تسلسل الإنسان البدائي المتقدم Neanderthal introgressed مقابل تسلسل المستنفد depleted sequences.

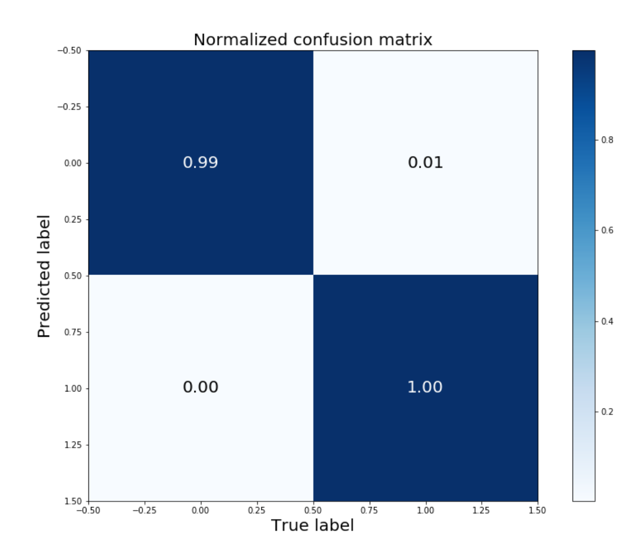
```
import itertools
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix

plt.figure(figsize = (15,10))

predicted_labels = model.predict(X_test)
cm = confusion_matrix(y_test, [np.round(i[0]) for i in
predicted_labels])
print('Confusion matrix:\n',cm)

cm = cm.astype('float') / cm.sum(axis = 1)[:, np.newaxis]
plt.imshow(cm, cmap = plt.cm.Blues)
plt.title('Normalized confusion matrix', fontsize = 20)
plt.colorbar()
plt.xlabel('True label',fontsize=20); plt.ylabel('Predicted
label',fontsize=20)
for i, j in itertools.product(range(cm.shape[0]),
range(cm.shape[1])):
    plt.text(j, i, format(cm[i, j], '.2f'),
             horizontalalignment = 'center', verticalalignment =
'center',
             fontsize = 20, color='white' if cm[i, j] > 0.5 else
'black')

scores = model.evaluate(X_test, y_test, verbose = 0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```



الآن لدينا نموذج LSTM مدرب سنستخدمه لاحقًا للتنبؤ بتسلسل الجينات الموروثة من إنسان نياندرتال. حان الوقت الآن لتفسير النموذج الذي يتضمن تصور المفردات والكشف عن معظم k-mers التنبؤية التي تقود التصنيف.

تصور تضمين الكلمة

في الوقت الحالي، يتم تمثيل كل k-mer / كلمة بمتجه 10 (10-dimensional vector) أبعاد نظرًا لأننا طبقنا طبقة تضمين Embedding Layerفي الواجهة الأمامية للشبكة. لتصور ما تعلمته طبقة التضمين، نحتاج أولاً إلى حفظ أوزان الطبقة وكلمات المفردات.

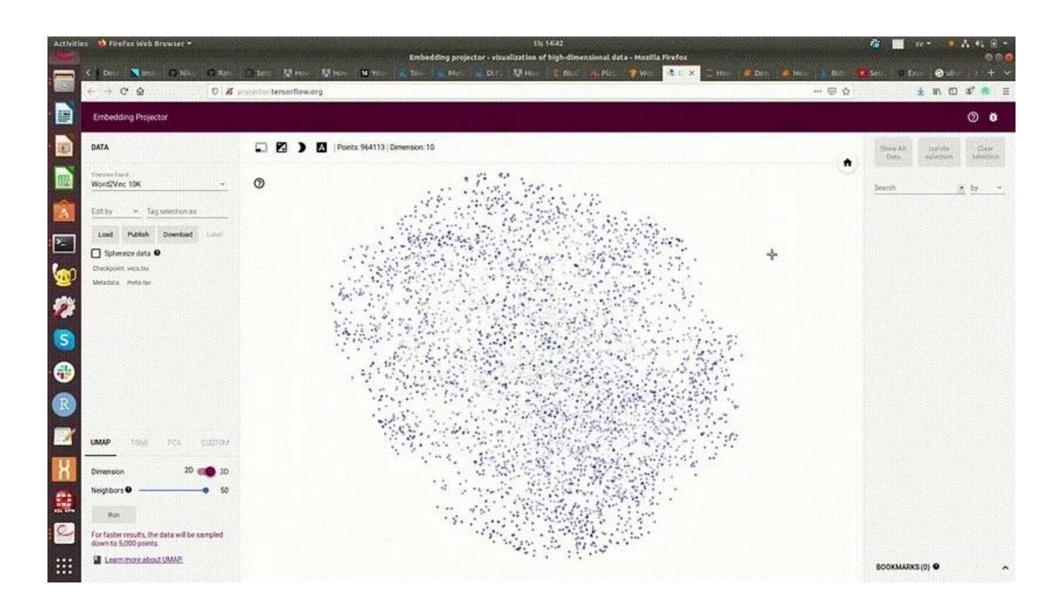
```
import io

e = model.layers[0]
weights = e.get_weights()[0]
words = [i.upper() for i in list(tokenizer.index_word.values())]

out_v = io.open('vecs.tsv', 'w', encoding='utf-8')
out_m = io.open('meta.tsv', 'w', encoding='utf-8')

for num, word in enumerate(words):
  vec = weights[num + 1] # skip 0, it's padding.
  out_m.write(word + "\n")
  out_v.write('\t'.join([str(x) for x in vec]) + "\n")
out_v.close()
out_m.close()
```

من أجل التصور، يمكننا استخدام Tensorflow Embedding Projector من هنا http://projector.tensorflow.org/ والتحقق من تجميع مجموعات k-mers باستخدام تقنية UMAP لتقليل الأبعاد غير الخطية.



بدت العلاقات بين k-mers التي تعلمتها طبقة التضمين مختلفة تمامًا عن تضمينات Word2Vec المقدمةفي المنشور السابق ولم تكشف عن أي تجمعات واضحة لـ k-mers الغنية بـ AT و GC الغنية.

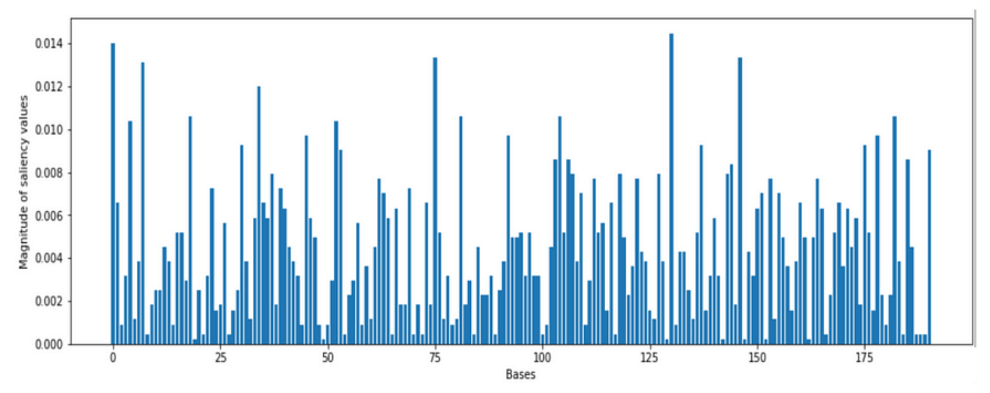
تحديد K-mers التنبؤية

تتمثل إحدى طرق بناء عناصر استيراد الميزات للشبكة العصبيةفي فرض بعض التقليب perturbation على بيانات الإدخال ومراقبة التباينفي دقة التنبؤ عند إخراج الشبكة. هنا، نود العثور على k-mers الأكثر إفادة، ومع ذلك،في حالتنا، كان لمصفوفة الإدخال X أبعاد (737340, 191)، حيث يمثل البعد الأول عدد أمثلة التدريب للشبكة العصبية، والبعد الثاني يتوافق مع عدد الكلماتفي كل جملة / تسلسل. هذا يعني أن فهرسًا index (أو موضعًا positionفي الجملة) لكل كلمة / k-mer كان ميزة لمصفوفة الإدخال X. لذلك إذا قمنا بتبديل كل ميزة feature من 191 ميزة، واحدة تلو الأخرى عبر 737340 عينة، وتحقق من الانخفاضفي الدقة مقارنة بمصفوفة الإدخال غير المقلبة un-perturbed input matrix، يمكننا ترتيب الميزات حسب أهميتها للتنبؤ النهائي.في حالتنا، فإن ميزات الترتيب تعادل تحديد أهم مواضع الكلمات / k-mers عبر جميع الجمل / التسلسلات.

```
import matplotlib.pyplot as plt

perm_scores = []
for i in range(X_test.shape[1]):
    X_test_perm = X_test
    X_test_perm[:,i] = random.sample(list(X_test[:,i]),
len(list(X_test[:,i])))
    perm_scores.append(abs(model.evaluate(X_test_perm, y_test,
verbose = 0)[1]*100 -
                              model.evaluate(X_test, y_test, verbose =
0)[1]*100))

plt.figure(figsize=[16,5])
barlist = plt.bar(np.arange(len(perm_scores)), perm_scores)
plt.xlabel('Bases'); plt.ylabel('Magnitude of saliency values')
```

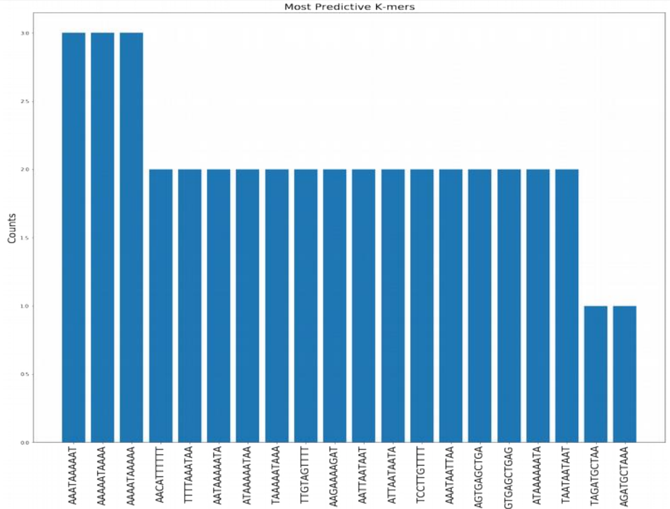


إذا حددنا مواضع positions / مؤشرات indices الكلمات التي تغير الدقة فوق 0.01، فإننا نصل إلى الكلمات 0 و 4 و 7 و 18 و 34 و 52 و 75 و 81 و 104 و 130 و 146 و 182 باعتبارها الأكثر أهمية. ربما نلاحظ بعض الإثراء للكلمات المهمةفي بداية الجمل. الآن نتحقق ببساطة من الكلمات الأكثر شيوعًا / k-mers عبر جميع الجملفي المواضع أعلاه.

```
import pandas as pd
from collections import Counter

scores_df = pd.DataFrame({'Base': range(len(perm_scores)),'Score':
perm_scores})
informative_indices = list(scores_df[scores_df['Score'] >
0.01].index)
reverse_word_map = dict(map(reversed,
tokenizer.word_index.items()))
def sequence_to_text(list_of_indices):
    return [reverse_word_map.get(letter) for letter in
list_of_indices]
my_texts = list(map(sequence_to_text, X_test[informative_indices,
:]))

fig = plt.figure(figsize = (20, 15))
D = dict(Counter([item.upper() for sublist in my_texts
                  for item in sublist]).most_common(20))
plt.bar(range(len(D)), list(D.values()), align = 'center')
plt.title('Most Predictive K-mers', fontsize = 20)
plt.ylabel("Counts", fontsize = 20); plt.xticks(rotation = 90)
plt.xticks(range(len(D)), list(D.keys()), fontsize = 20)
```



مما يبعث على الاطمئنان، نلاحظ أن k-mers الغني بـ AT (AT-rich k-mers) يميزون أكثر ما يميزون بين تسلسلات النياندرتال المتقدمة والمستنفدة كما تم عرضه أيضًافي المنشور السابق.

توقع جينات النياندرتال

الآن سننظرفي تسلسل الجينات البشرية ونموذج LSTM المدرب من أجل عمل تنبؤات عن مدى احتمالية توريث كل جين بشري من إنسان نياندرتال.في المنشور السابق، أوضحت كيفية استخراج تسلسل الجيناتفي ملف فاستا fasta-file منفصل، لذلك سنقرأ الآن هذه التسلسلات، ونقسمها إلى k-mers، ونحول k-mers إلى أعداد صحيحة باستخدام Tokenizer، ونتوقع أصل إنسان نياندرتال باستخدام نموذج LSTM المدرب.

```
import pandas as pd
from Bio import SeqIO
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences

e = 0; gene_seqs = []; gene_ids = []
for gene in SeqIO.parse('hg19_gene_clean.fa', 'fasta'):
    cutoff = 200
    if len(str(gene.seq)) < cutoff:
        continue
    gene_ids.append(str(gene.id)); s_gene = str(gene.seq)[0:cutoff]
    gene_seqs.append(s_gene)
    e = e + 1
    if e%10000 == 0:
        print('Finished ' + str(e) + ' genes')

def getKmers(sequence, size):
    return [sequence[x:x+size].upper() for x in range(len(sequence)
- size + 1)]

kmer = 10
gene_texts = [' '.join(getKmers(i, kmer)) for i in gene_seqs]

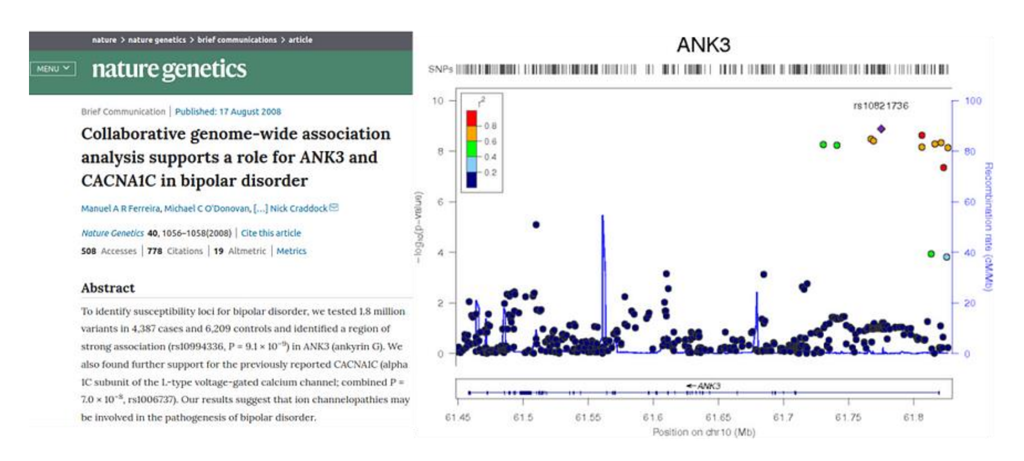
tokenizer = Tokenizer(); tokenizer.fit_on_texts(gene_texts)
encoded_docs = tokenizer.texts_to_sequences(gene_texts)
max_length = max([len(s.split()) for s in gene_texts])
X_gene = pad_sequences(encoded_docs, maxlen = max_length, padding =
'post')

gene_predictions = model.predict_classes(X_gene)
gene_predictions_prob = model.predict_proba(X_gene)

gene_pred_df = pd.DataFrame({'Gene': gene_ids, 'Gene_Symbol':
gene_symbol,
                             'Predict':
list(gene_predictions.flatten()),
                             'Prob':
list(gene_predictions_prob.flatten())})
gene_pred_df = gene_pred_df.sort_values(['Prob'], ascending =
False)
gene_pred_df[(gene_pred_df['Predict'] == 1) & (gene_pred_df['Prob']
> 0.8)]
```

	Gene	Gene_Symbol	Predict	Prob
7360	chr12:120884241-120901556	GATC	1	1.000000
23845	chr6:73844526-73853237	KCNQ5-AS1	1	1.000000
20238	chr4:40194587-40246384	RHOH	1	1.000000
2022	chr1:174769035-174964445	RABGAP1L	1	1.000000
2024	chr1:174904084-174923398	LOC101928696	1	1.000000
8518	chr14:75230069-75304013	YLPM1	1	1.000000
2068	chr1:180199433-180244188	LHX4	1	1.000000
8408	chr14:61176256-61190852	SIX4	1	1.000000
2107	chr1:186369704-186370587	OCLM	1	1.000000
2114	chr1:187412760-187446354	LINC01037	1	1.000000
8219	chr14:24641234-24649463	REC8	1	1.000000
8198	chr14:24422944-24438488	DHRS4	1	1.000000
8109	chr14:20779527-20801471	CCNB1IP1	1	1.000000
2204	chr1:202955580-202976393	LOC100506747	1	1.000000
8033	chr13:111766159-111768025	ARHGEF7-AS2	1	1.000000
7926	chr13:85937738-86118797	LINC00351	1	1.000000
26185	chr7:33019086-33046543	FKBP9	1	1.000000
7701	chr13:39106137-39260812	LINC00437	1	1.000000
2418	chr1:226335704-226342678	ACBD3-AS1	1	1.000000
7465	chr12:125396191-125399587	UBC	1	1.000000
29638	chrUn_gl000211:48503-93165	FLJ43315	1	1.000000
18818	chr3:99979661-100044096	TBC1D23	1	1.000000
17116	chr20:45947246-45949498	LOC100131496	1	1.000000
8943	chr15:45406523-45410301	DUOXA2	1	1.000000
8994	chr15:55647421-55700708	CCPG1	1	1.000000
31193	chrX:154718673-154842622	TMLHE	1	1.000000

عند التحقق من عدد الجينات المتوقع أن تكون موروثة من إنسان نياندرتال، نؤكد النتيجة من المنشور السابق أن الغالبية العظمى من الجينات، أي 22000 من أصل 31000 جينة بشرية تم تحليلها، خالية من أصل إنسان نياندرتال. لذا توصلنا مرة أخرى إلى استنتاج مفاده أن التطور لسبب ما دفع بأصول إنسان نياندرتال بعيداً عن الجينات التي تعد معظم العناصر الوظيفية للجينوم البشري. بإلقاء نظرة فاحصة على قائمة جينات النياندرتال "الناجية survived " ومقارنتها مع تنبؤات Random Forest ضمن نموذج Bag of Words من المنشور السابق، نلاحظ عددًا من الجينات التي لها ارتباطات مع سمات وأمراض بشرية مختلفة. على سبيل المثال، تم التنبؤ بجين ANK3 باحتمال> 99٪ من قبل كل من LSTM وRandom Forest ليكون من أصل إنسان نياندرتال. من المعروف أن هذا الجين مرتبط بالاضطراب ثنائي القطب Bipolar Disorder ويتم التعبير عنهفي الغالبفي دماغ الإنسان.


nature > nature genetics > brief communications > article

MENU

nature genetics

Brief Communication | Published: 17 August 2008

Collaborative genome-wide association analysis supports a role for ANK3 and CACNA1C in bipolar disorder

Manuel A R Ferreira, Michael C O'Donovan, [...] Nick Craddock

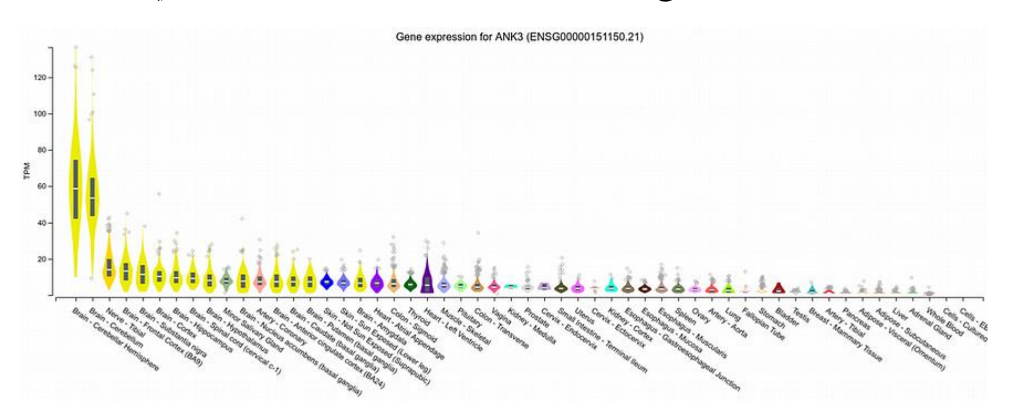
Nature Genetics **40**, 1056–1058(2008) | Cite this article

508 Accesses | **778** Citations | **19** Altmetric | Metrics

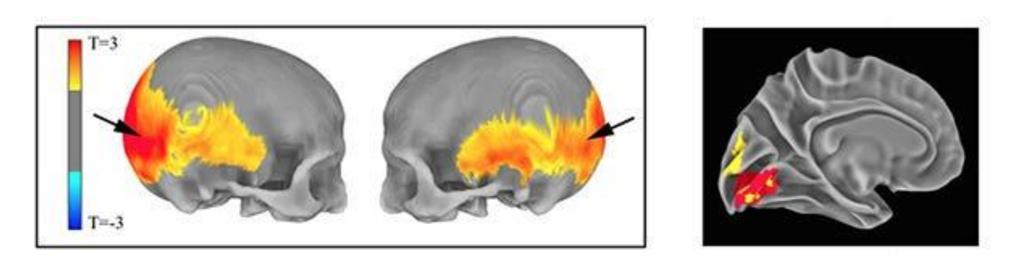
Abstract

To identify susceptibility loci for bipolar disorder, we tested 1.8 million variants in 4,387 cases and 6,209 controls and identified a region of strong association (rs10994336, $P = 9.1 \times 10^{-9}$) in ANK3 (ankyrin G). We also found further support for the previously reported CACNA1C (alpha 1C subunit of the L-type voltage-gated calcium channel; combined $P = 7.0 \times 10^{-8}$, rs1006737). Our results suggest that ion channelopathies may be involved in the pathogenesis of bipolar disorder.

يرتبط جين ANK3 المتوقع أن يكون من أصل إنسان نياندرتال بالاضطراب ثنائي القطب



بالإضافة إلى ذلك، اقترحت بعض الدراسات أن أجزاء من أدمغة الإنسان المتورطةفي الاضطرابات النفسية قد تتشكل من خلال الوراثة الجينية من إنسان نياندرتال.



يؤوي الدماغ البشري "الصدى المتبقي residual echo " لجينات النياندرتال

هذا هو المكان الذي يمكن لعلم التطور أن يطلع فيه الطب الحيوي Biomedicine على التطور التاريخي وأصل السمات التي أدت إلى أمراض الإنسان الحديثة.

الاستنتاج

في هذا المنشور، تعلمنا أنه قد يكون من المفيد تطبيق نماذج الذاكرة الطويلة مثل LSTM على بيانات تسلسل الحمض النووي DNA المعروف بامتلاكها لارتباطات طويلة المدى على طول التسلسلات. لقد أظهرنا أن LSTM حقق دقة مذهلة وتفوق في الأداء على جميع النماذج الأخرى التي تكشف عن مناطق تقدم الإنسان البدائي في الجينوم البشري الحديث. كشف تفسير النموذج أن k-mers الغني بـ AT هو الفرق الرئيسي بين الإنسان البدائي وزخارف الحمض النووي البشري الحديث. لقد توقعنا أن تكون غالبية الجينات البشرية خالية من أصل إنسان نياندرتال مما يعني ضمناً اختيارًا سلبيًا قويًا خلال التطور ضد أصل إنسان نياندرتال. يرتبط عدد من الجينات التي يُتوقع أن تكون موروثة من إنسان نياندرتال بسمات بشرية مثيرة للاهتمام مثل الاضطراب ثنائي القطب.

المصدر:

https://towardsdatascience.com/lstm-to-detect-neanderthal-dna-843df7e85743

15) التعلم العميق على الميكروبيوم البشري Deep Learning on Human Microbiome

(استنتاج التركيب الميكروبي لعينة من تسلسل الحمض النووي)

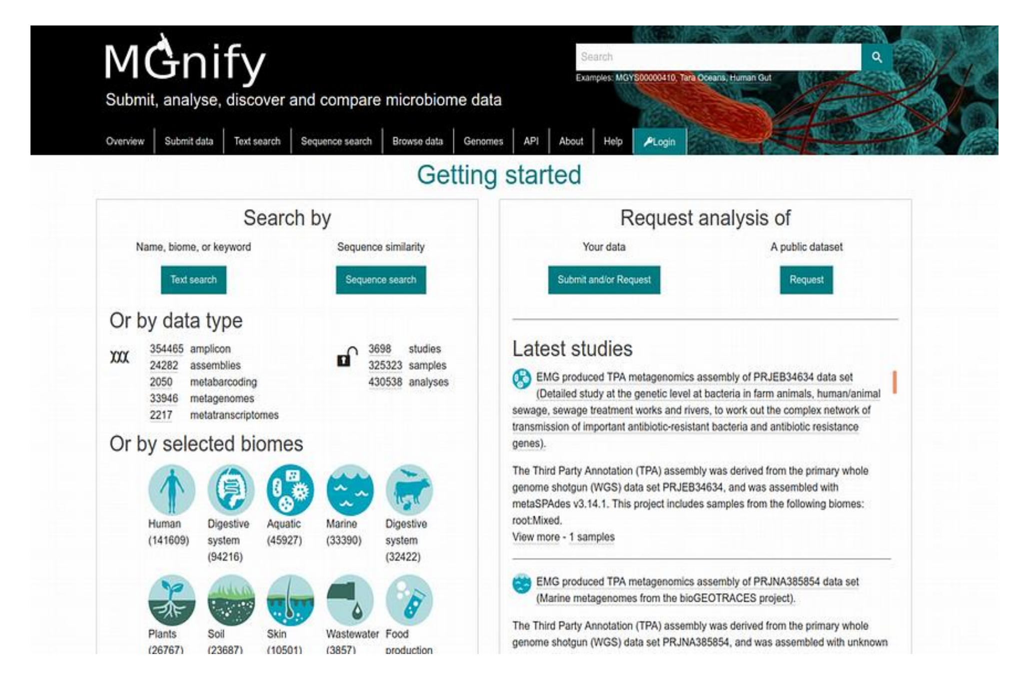
إنه المنشور التاسع في عمودي "التعلم العميق لعلوم الحياة Deep Learning for Life Sciences" حيث أحاول عرض أمثلة ملموسة لتطبيق الشبكات العصبية الاصطناعية artificial neural networks على مشاريع العالم الحقيقي من البيولوجيا الحاسوبية Computational Biology وعلوم الحياة Life Sciences. في السابق، قمنا بتغطية بعض تطبيقات التعلم العميق للحمض النووي القديم Ancient DNA، وبيولوجيا الخلية المفردة Single Cell Biology، وتكامل البيانات Data Integration، والتشخيص السريري Microscopy Imaging، والتصوير المجهري Microscopy Imaging بالإضافة إلى بعض التقنيات من معالجة اللغة الطبيعية (NLP) لاستنتاج مناطق إدخال الإنسان البدائي في الجينوم البشري الحديث. في هذه المقالة، سوف نتعلم لماذا يمثل الميكروبيوم البشري Human Microbiome البيانات الضخمة في علم الأحياء Biology، وكيفية تدريب مصنف الشبكة العصبية التلافيفية Convolutional Neural Network (CNN) للتنبؤ بالتركيب الميكروبي لعينة الميتاجينوميات metagenomics sample بدءًا من التسلسلات الأولية في ملف fastq.

الميتاجينوميات كبيانات كبيرة

لسوء الحظ، ليس من السهل البدء في تطوير نماذج التعلم الآلي للتطبيقات البيولوجية والطبية الحيوية، في رأيي، بسبب نقص البيانات. ناقشت بعض التحديات في رسالتي السابقة هل لدينا بيانات كبيرة في علوم الحياة؟ لقد ذكرت هناك ثلاث اتجاهات في علوم الحياة تبدو واعدة للذكاء الاصطناعي والتعلم الآلي لأنه يبدو أن لديهم كميات كافية من البيانات، وهي:

1. أوميكس خلية مفردة Single Cell Omics.
2. التصوير المجهري Microscopy imaging.
3. بيانات الجينوميات Genomics data تعتبر التسلسل ملاحظات إحصائية.

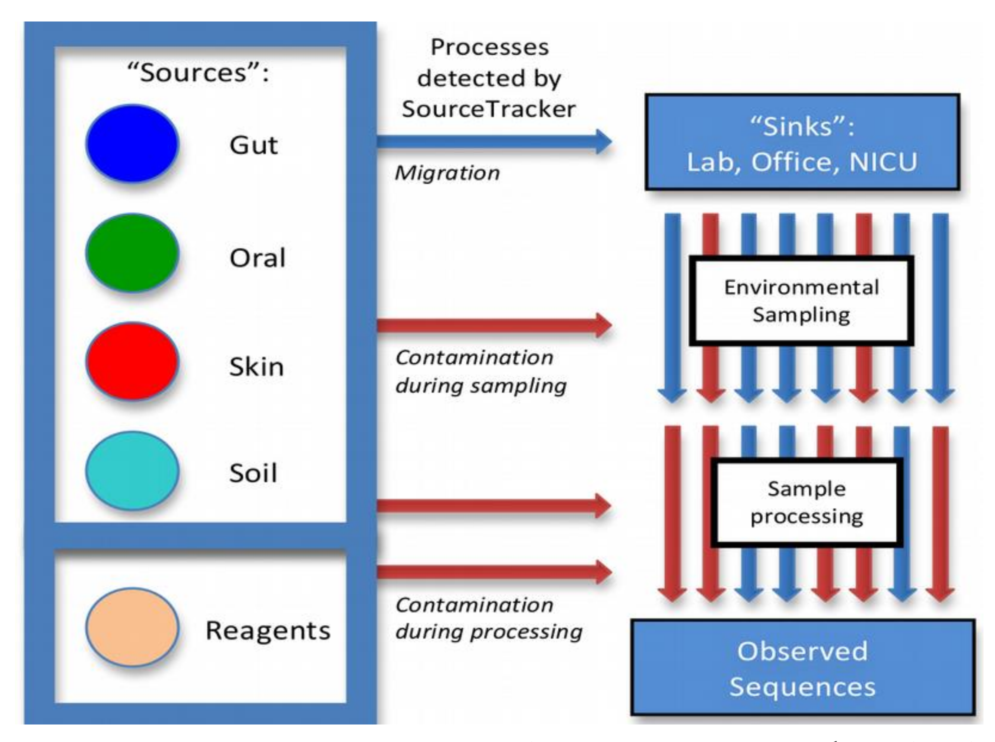
لاحقًا في التعليقات اقترح Mikael Huss بيانات الميتاجينوميات Metagenomics والميكروبيوم Microbiome باعتبارها المصدر الرابع للبيانات الكبيرة. في الواقع، يمكن للمرء أن يجد الكثير من بيانات الميتاجينوميات المتاحة للجمهور، على سبيل المثال، باستخدام مورد EMBL-EBI الرائع MGnify.



السبب وراء تسمية Metegenomics ببيانات كبيرة Big Data هو تسلسل البندقية shotgun sequencing الرخيص نسبيًا، وبالتالي يتم ترتيب آلاف العينات، والطبيعة المنخفضة الأبعاد نسبيًا لبيانات الميتاجينوميات.في الواقع، يؤدي رسم خرائط لشظايا الحمض النووي DNA fragments المتسلسلة للكائنات الميكروبية microbial organisms المعروفة عادةً إلى اكتشاف 500ـ1000 ميكروب موثوق به، وهي ليستفي الحقيقة بيانات عالية الأبعاد مقارنةً، على سبيل المثال، بالتنوع الجيني genetic variation أو بيانات المثيلة methylation التي تحتوي على ملايين من الأبعاد.

التركيب الميكروبي مع SourceTracker

مع الأخذفي الاعتبار أن Metagenomics توفر الكثير من البيانات لتطوير خوارزميات الآلة والتعلم العميق، سأحاولفي هذه المقالة تدريب شبكة عصبية تلافيفية (CNN) يمكنها تقدير مصادر المجتمعات الميكروبية لعينة معينة. على سبيل المثال، أخذ عينة مسحة swab من سطح المكتبفي مكتبي، ما نوع الميكروبات التي أتوقع أن أجدها هناك؟ هل تشبه جراثيم بشرتي، أو ربما ميكروبات فموية من فمي؟ يمكن تقدير ذلك باستخدام برنامج رائع يسمى SourceTracker.

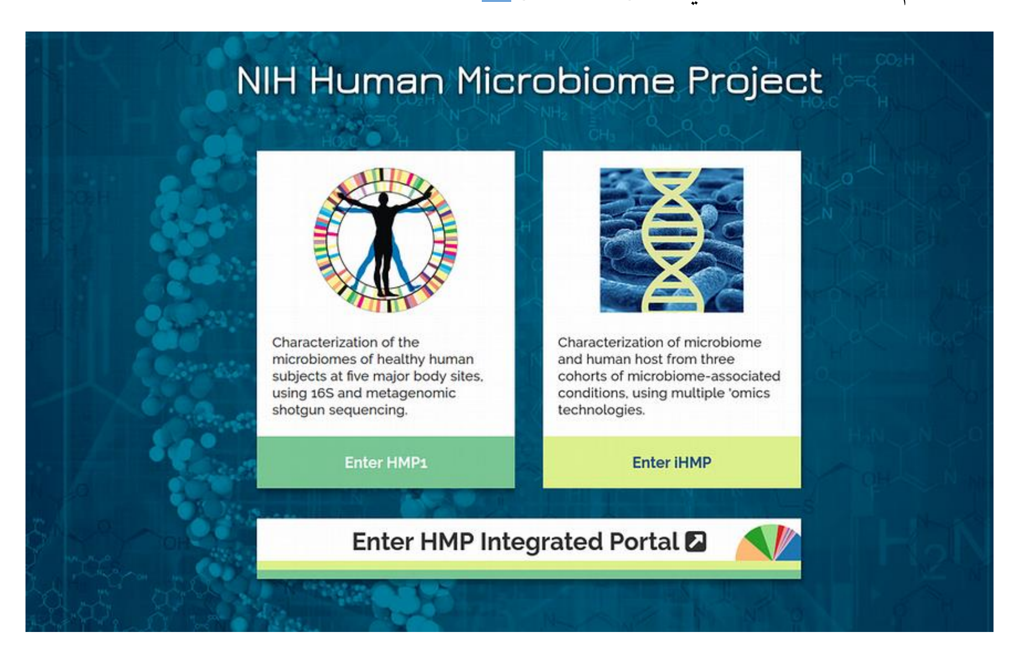


باختصار، SourceTracker هي نسخة بايزية Bayesian version من خوارزمية تجميع clustering algorithm نموذج خليط غاوسي Gaussian Mixture Model (GMM) (تبدو مشابهة لخليط ديريكليت متعدد الحدود Direchlet Multinomial Mixtures) التي يتم تدريبها على مجموعة بيانات مرجعية تسمى المصادر Sources، أي فئات مختلفة مثل التربة أو جرثومة الفم البشري أو الأمعاء البشرية المجتمعات وما إلى ذلك، ويمكن تقدير نسبة / مساهمة كل مصدرفي عينات اختبار تسمى الأحواض Sinks. لذلك، تقوم خوارزمية SourceTracker بشكل صريح بنمذجة عينة حوض Sink sample كمزيج من المصادر. تأتي نكهة Bayesian للخوارزمية من استخدام Direchlet priors عند ملاءمة fitting النموذج.

في الأصل، تم تطوير SourceTracker لبيانات 16S، أي باستخدام جينات RNA الريبوسوم 16S فقط.في المقابل، سنحاول هنا تدريب المصنف باستخدام بيانات ميتاجينوميات البندقية shotgun metagenomics. بالإضافة إلى ذلك، لم يتم تصميم SourceTracker لاستخدامهفي تسلسل الميتاجينوميات الخام، ولكنه يحتاج بدلاً من ذلك إلى وفرة ميكروبية microbial abundances (وفرة OTU) محددة بطريقة ما، على سبيل المثال من خلال خط أنابيب QIIME أو MetaPhlan2 أو Kraken2.في المقابل، فإن الهدف من مصنف CNN الذي سنقوم بتدريبه هنا هو الانتقال من ملف fastq إلى التنبؤ بالتركيب الميكروبي لعينة metagenomic دون تحديد الكميات الوفرة الميكروبية.

بيانات مشروع الميكروبيوم البشري (HMP)

لتدريب مصنف CNN، سنستخدم مورد بيانات مشروع الميكروبيوم البشري العظيم Human Microbiome Project (HMP). يوفر HMP وفرة من الأصناف الميتاجينومية المحددة كمياً باستخدام MetaPhlan2 والتي يمكن تنزيلها من هنا.



تم حساب ملامح الميتاجينوميات لأربعة أنسجة بشرية (مجتمعات ميكروبية microbial communities): الفم Oral، الأمعاء Gut، الجلد Skin، المهبل Vagina. أدناه، سنستخدم مصفوفة الوفرة الميكروبية microbial abundance matrix لاختيار الأجناس البكتيرية الخاصة بالأنسجة tissue-specific bacterial genera، أي مجموعات البكتيريا التي تتوفر بكثرةفي واحد فقط من الأنسجة الأربعة، وبالتالي يمكن أن تكون بمثابة صانعي الميكروبات للأنسجة البشرية. لاحقًا، سوف نستخدم الجينومات المرجعية للأجناس البكتيرية الخاصة بالأنسجة لتدريب مصنف CNN على التعرف على المجتمعات الميكروبية (الفم، والأمعاء، والجلد، والمهبل) من التسلسلات الأوليةفي ملف fastq. سنبدأ بقراءة مصفوفة الوفرة الميكروبية وتصور العينات من الأنسجة البشرية الأربعة عبر مخططات تقليل الأبعاد (PCA و tSNE).

```
import pandas as pd
import seaborn as sns
from umap.umap_ import UMAP
import matplotlib.pyplot as plt
from sklearn.manifold import TSNE
from sklearn.decomposition import PCA

sns.set(font_scale = 1.5)
```

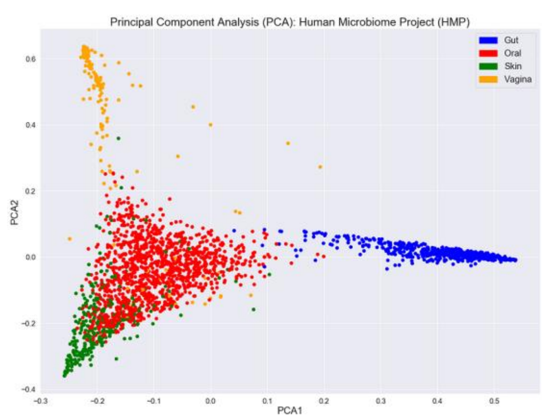
```
microb = pd.read_csv('microb.txt', sep = '\t')
phen = pd.read_csv('phen.txt', sep = '\t')
phen['Color'] = 'blue'
phen['Color'][phen['STArea'] == 'Oral'] = 'red'
phen['Color'][phen['STArea'] == 'Skin'] = 'green'
phen['Color'][phen['STArea'] == 'Vaginal'] = 'orange'

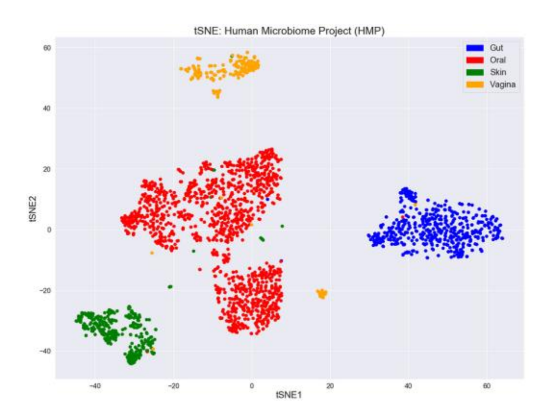
X = microb.values
X = np.log10(X + 1)

plt.figure(figsize = (20, 15))
X_reduced = PCA(n_components = 2).fit_transform(X)
plt.scatter(X_reduced[:, 0], X_reduced[:, 1], s = 50, c =
phen['Color'])
plt.title('Principal Component Analysis (PCA): Human Microbiome
Project (HMP)',
          fontsize = 25)
plt.xlabel("PCA1", fontsize = 22); plt.ylabel("PCA2", fontsize =
22)

from matplotlib import cm
import matplotlib.patches as mpatches
my_legends = [mpatches.Patch(color = 'blue', label = 'Gut'),
              mpatches.Patch(color = 'red', label = 'Oral'),
              mpatches.Patch(color = 'green', label = 'Skin'),
              mpatches.Patch(color = 'orange', label = 'Vagina')]
plt.legend(handles = my_legends, fontsize = 20)

plt.figure(figsize = (20, 15))
model = TSNE(learning_rate = 200, n_components = 2, random_state =
123,
             perplexity = 50, init = X_reduced, n_iter = 1000,
verbose = 2)
tsne = model.fit_transform(X)
plt.scatter(tsne[:, 0], tsne[:, 1], s = 50, c = phen['Color'])
plt.title('tSNE: Human Microbiome Project (HMP)', fontsize = 25)
plt.xlabel("tSNE1", fontsize = 22); plt.ylabel("tSNE2", fontsize =
22)
plt.legend(handles = my_legends, fontsize = 20)
```





في كل من مخططات PCA و tSNE، يمكننا أن نرى أن العينات يمكن فصلها بوضوح بواسطة الأنسجة الأصلية من حيث الوفرة الميكروبية. هذا يعني أنه يجب أن يكون من السهل جداً تدريب مصنف CNN يمكنه التنبؤ بنسيج المنشأ لعينة معينة عن طريق فحص تسلسل الحمض النووي الميكروبي microbial DNA sequencesفي العينة. ومع ذلك، من أجل صنع مخططات PCA وtSNE، استخدمنا الوفرة الميكروبية microbial abundances وليس البيانات المتسلسلة نفسها، لذلك سيتعين علينا أولاً استخراج الكائنات الميكروبية التي تفصل أفضل عينات الأمعاء والفم والجلد والمهبل. سيتم استخدام الجينومات المرجعية Reference genomes من الكائنات الميكروبية الأكثر تمييزاًفي وقت لاحق لإنشاء مجموعة بيانات التدريب train data set من التسلسلات لمصنف CNN.

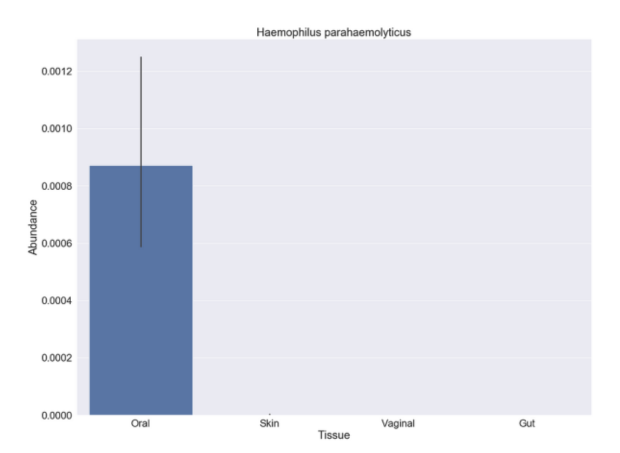
اختيار الميكروبات الخاصة بالأنسجة

لاختيار الكائنات الميكروبية التي تفصل بشكل أفضل بين عينات الأمعاء، والفم، والجلد، والمهبل، يمكننا على سبيل المثال تدريب مُصنف Random Forest وإلقاء نظرة على أهمية الميزات واتجاهات تأثيرها. ومع ذلك، يبدو أن هناك الكثير من الميكروبات الخاصة بالأنسجة، أي الميكروبات التي لها وفرة عاليةفي نسيج واحد ولكنها تكاد تكون معدومةفي جميع الأنسجة الأخرى. على سبيل المثال، إذا نظرنا إلى Haemophilus parahaemolyticus الذي يحتوي على مؤشر 1950في إطار بيانات الوفرة الميكروبية، فإنه يبدو خاص بالفم very Oral-specific:

```
ind = 1950; groupped_data =
pd.DataFrame({'Tissue':
                list(['Oral']*phen['STArea'].value_counts()['Oral'] +
                     ['Skin']*phen['STArea'].value_counts()['Skin'] +

['Vaginal']*phen['STArea'].value_counts()['Vaginal'] +
                     ['Gut']*phen['STArea'].value_counts()['Gut']),
                'Abundance':
                list(microb.loc[phen['ID'][phen['STArea'] ==
'Oral'],].iloc[:, ind]) +
```

```
                 list(microb.loc[phen['ID'][phen['STArea'] ==
'Skin'],].iloc[:, ind]) +
                 list(microb.loc[phen['ID'][phen['STArea'] ==
'Vaginal'],].iloc[:,ind])+
                 list(microb.loc[phen['ID'][phen['STArea'] ==
'Gut'],].iloc[:, ind])})
plt.figure(figsize = (20, 15))
sns.barplot(x = "Tissue", y = "Abundance", data = groupped_data)
plt.title('Haemophilus parahaemolyticus')
```



أدناه، من أجل التبسيط، سنركز على الميكروبات البكتيرية فقط، أي سيتم تجاهل العتائق archaea والفيروسات viruses، وسيتم الاحتفاظ بالكائنات البكتيرية المصنفة على مستوى الجنس في مجموعة بيانات HMP لمزيد من التحليل. سنقوم ببناء قوائم بالأجناس البكتيرية الخاصة بالأنسجة tissue-specific bacterial genera، ونستخدم أسمائها "لالتقاط "grepping الجينومات المرجعية البكتيرية من قاعدة بيانات RefSeq NCBI. لذلك، قمنا هنا بتقسيم المصفوفة الإجمالية للوفرة الميكروبية microbial abundances إلى وفرة الأجناس البكتيرية bacterial genera abundances فقط، يمكننا أن نرى أن هناك 227 جنسًا بكتيريًا موجودًافي مشروع HMP. بعد ذلك، سنستعرض جميع الأجناس البكتيرية HMP ونبني قوائم بالأجناس الخاصة بالفم والجلد والمهبل والأمعاء وفقًا لمعيار بسيط: إذاكان الجنس البكتيري أكثر وفرة 10 مرات في نسيج واحد مقارنةً بالجميع الأنسجة الأخرى، يعتبر الجنس خاصًا بالأنسجة tissue-specific.

```
microb = microb[[i for i in list(microb.columns) if not 's__' in i
and 'g__' in i
                    and 'k__Bacteria' in i and not 'noname' in i
                    and not 'unclassified' in i]]

oral_specific_list = list(); gut_specific_list = list();
skin_specific_list = list(); vagina_specific_list = list()

for ind in range(len(microb.columns)):
```

```
    my_data = pd.DataFrame({'Tissue':
      list(['Oral']*phen['STArea'].value_counts()['Oral'] +
      ['Skin']*phen['STArea'].value_counts()['Skin'] +
      ['Vaginal']*phen['STArea'].value_counts()['Vaginal'] +
      ['Gut']*phen['STArea'].value_counts()['Gut']),

'Abundance':list(microb.loc[phen['ID'][phen['STArea']=='Oral'],].il
oc[:,ind])+
      list(microb.loc[phen['ID'][phen['STArea'] ==
'Skin'],].iloc[:, ind]) +
      list(microb.loc[phen['ID'][phen['STArea'] ==
'Vaginal'],].iloc[:, ind]) +
      list(microb.loc[phen['ID'][phen['STArea'] == 'Gut'],].iloc[:,
ind])})

    ratio = float(my_data.groupby('Tissue',
sort=False).mean().sort_values(
        'Abundance', ascending = True).iloc[3, :] /
my_data.groupby('Tissue',
        sort = False).mean().sort_values('Abundance', ascending =
True).iloc[2, :])

    if(ratio > 10 and 'k__Bacteria' in microb.columns[ind]
       and 'g__' in microb.columns[ind]):

        if(my_data.groupby('Tissue', sort =
False).mean().sort_values(
        'Abundance', ascending = True).index[3] == 'Oral'):
            oral_specific_list.append(microb.columns[ind])

        if(my_data.groupby('Tissue', sort =
False).mean().sort_values(
        'Abundance', ascending = True).index[3] == 'Skin'):
            skin_specific_list.append(microb.columns[ind])

        if(my_data.groupby('Tissue', sort =
False).mean().sort_values(
        'Abundance', ascending = True).index[3] == 'Vaginal'):
            vagina_specific_list.append(microb.columns[ind])

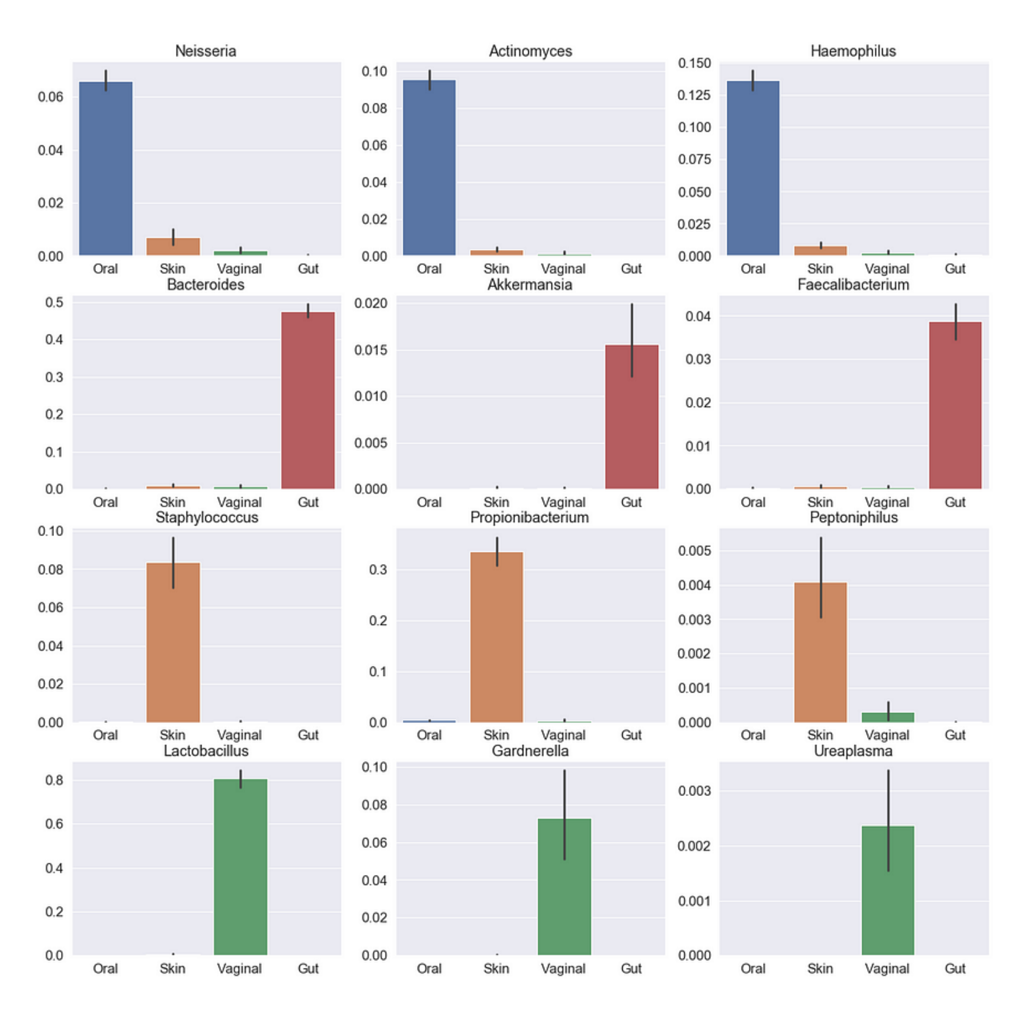
        if(my_data.groupby('Tissue', sort =
False).mean().sort_values(
        'Abundance', ascending = True).index[3] == 'Gut'):
            gut_specific_list.append(microb.columns[ind])

tissue_specific_list = []
for tissue in [oral_specific_list, gut_specific_list,
skin_specific_list,
               vagina_specific_list]:
    parsed_genus_bacteria = list(set([i.split('|')[5] for i in
tissue]))
    parsed_genus_bacteria = [i.replace('g__','') for i in
parsed_genus_bacteria]
    tissue_specific_genera = list(set(parsed_genus_bacteria))
    tissue_specific_list.append(tissue_specific_genera)
```

```
for index, tissue in enumerate(['oral','gut','skin','vagina']):
    with open('tissue_specific_genera_names_' + tissue +
'_full_list.txt', 'w') as f:
        for item in tissue_specific_list[index]:
            f.write("%s\n" % item)
```

يمنحنا هذا المعيار البسيط لخصوصية الأنسجة tissue-specificity 61 جنسًا خاصًا بالفم، و53 خاصًا بالأمعاء، و49 خاصًا بالجلد، و16 جنسًا بكتيريًا خاصًا بالمهبل. دعونا نختار ونتصور وفرة من بعض الأجناس الخاصة بالأنسجة من كل من الأنسجة الأربعة:

```
sample_microb_names = ['Neisseria', 'Actinomyces', 'Haemophilus',
                       'Bacteroides', 'Akkermansia',
'Faecalibacterium',
                       'Staphylococcus', 'Propionibacterium',
'Peptoniphilus',
                       'Lactobacillus', 'Gardnerella',
'Ureaplasma']
genus_index=[microb.columns.get_loc([i for i in microb.columns if
('g__'+j) in i][0])
               for j in sample_microb_names]
for i, ind in enumerate(genus_index):
    plt.subplot(4, 3, i + 1)
    groupped_data = pd.DataFrame({'Tissue':
      list(['Oral']*phen['STArea'].value_counts()['Oral'] +
      ['Skin']*phen['STArea'].value_counts()['Skin'] +
      ['Vaginal']*phen['STArea'].value_counts()['Vaginal'] +
      ['Gut']*phen['STArea'].value_counts()['Gut']),
      'Abundance': list(microb.loc[phen['ID'][phen['STArea'] ==
'Oral'],].iloc[:,ind])+
      list(microb.loc[phen['ID'][phen['STArea'] ==
'Skin'],].iloc[:, ind]) +
      list(microb.loc[phen['ID'][phen['STArea'] ==
'Vaginal'],].iloc[:, ind]) +
      list(microb.loc[phen['ID'][phen['STArea'] == 'Gut'],].iloc[:,
ind])})
    sns.barplot(x = "Tissue", y = "Abundance", data =
groupped_data)
    plt.xlabel(""); plt.ylabel("");
plt.title(sample_microb_names[i])
```



أعلاه، يمكننا أن نرى بوضوح أننا تمكنا من اختيار علامات أجناس بكتيرية قوية جداً خاصة بالأنسجة. المهبل كان الأقل، أي 16 جنساً خاصاً بالأنسجة. ومع ذلك، عند إلقاء نظرة فاحصة، يمكن للمرء أن يرى أنه اشتمل على أجناس مثل Burkholderia و Ralstonia و Bordetella المشبوهة ومن المحتمل جداً أن تكون ملوثات كاشف PCR (PCR reagent contaminants). لذلك قمنا بفحص قائمة الأجناس البكتيرية الـ 16 الخاصة بالمهبل بعناية ويمكننا أن نستنتج بثقة خصوصية المهبل vaginal specificity لـ 12 منهم فقط لأسباب مختلفة. لذلك، من أجل الحفاظ على التوازن بين الأنسجة لتحليل التصنيف، سنختار أكثر 12 علامة خاصة بالأنسجة لكل من الأنسجة الأربعة، وهنا تأتي الأجناس البكتيرية التي اخترناها:

	Oral	Gut	Skin	Vagina
0	Neisseria	Blautia	Staphylococcus	Mobiluncus
1	Veillonella	Faecalibacterium	Peptoniphilus	Sphingopyxis
2	Actinomyces	Bacteroides	Citrobacter	Ureaplasma
3	Haemophilus	Dorea	Enhydrobacter	Caulobacter
4	Rothia	Akkermansia	Finegoldia	Gardnerella
5	Leptotrichia	Clostridium	Propionibacterium	Chlamydia
6	Cardiobacterium	Ruminococcus	Acinetobacter	Asticcacaulis
7	Capnocytophaga	Subdoligranulum	Massilia	Mycobacterium
8	Oribacterium	Oxalobacter	Hymenobacter	Herbaspirillum
9	Alloprevotella	Oscillibacter	Corynebacterium	Lactobacillus
10	Gemella	Eubacterium	Bacillus	Achromobacter
11	Fusobacterium	Bilophila	Micrococcus	Atopobium

بعد الكثير من التفكير، قررت استبعاد Streptococcus من قائمة الأجناس الخاصة بالفم. على الرغم من أن جنس Streptococcus يبدو أنه خاص بالفم، إلا أنه يشمل الأنواع التي يمكن أن تكون وفيرةً في الأنسجة الأخرى مثل الأمعاء أيضًا. هذا يمكن أن يخلط بين مصنف CNN لأننا إذا قمنا بالإشارة إلى الجينومات الخاصة بالعقدية Streptococcus، فسننتهي بالعديد من التسلسلات "الملوثة contaminating" التي تنشأ من أنواع العقدية الخاصة بالأمعاء، ومع ذلك نحصل على تسميات فموية Oral labels خاطئة إذا افترضنا أن العقدية هي نوع فموي محدد oral-specific genus.

يمكننا التحقق من أن الأجناس الميكروبية المختارة يمكنها فصل عينات الفم، الأمعاء، الجلد، المهبل بشكل جيد. لهذا الغرض، دعونا نجري تجميعًا سريعًا استنادًا إلى مسافة ارتباط سبيرمان Spearman correlation distance بين العينات واستخدام 48 جنسًا ميكروبيًا محددًا (12 لكل من الأنسجة الأربعة).

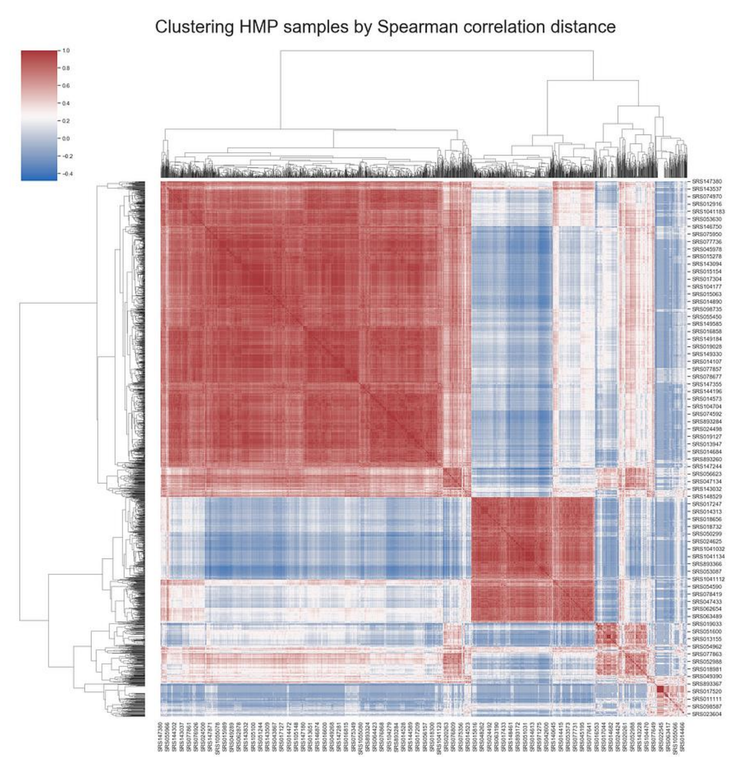
```
import scipy as sp
import seaborn as sns

sns.set(font_scale = 1)

microb.columns = [i.split('|')[5] for i in microb.columns]
microb.columns = [i.replace('g__','') for i in microb.columns]
```

```
gen_list = ['Neisseria', 'Veillonella', 'Actinomyces',
'Haemophilus', 'Rothia',
'Leptotrichia', 'Cardiobacterium',
'Capnocytophaga','Oribacterium','Alloprevotella',
'Fusobacterium', 'Blautia', 'Faecalibacterium',
'Bacteroides','Dorea','Akkermansia',
'Clostridium', 'Ruminococcus', 'Subdoligranulum', 'Oxalobacter',
'Oscillibacter',
'Eubacterium', 'Bilophila', 'Staphylococcus', 'Peptoniphilus',
'Citrobacter',
'Enhydrobacter', 'Finegoldia', 'Propionibacterium',
'Acinetobacter', 'Massilia',
'Hymenobacter', 'Corynebacterium', 'Bacillus', 'Micrococcus',
'Mobiluncus',
'Sphingopyxis','Ureaplasma','Caulobacter','Gardnerella','Chlamydia'
,'Asticcacaulis',
'Mycobacterium', 'Herbaspirillum', 'Lactobacillus',
'Achromobacter', 'Atopobium']
microb_genus_selected = microb[gen_list]

rho, _ = sp.stats.spearmanr(microb_genus_selected.T)
rho = pd.DataFrame(rho, index = microb_genus_selected.index,
                   columns = microb_genus_selected.index)
sns.clustermap(rho, cmap = 'vlag', figsize = (20, 20))
plt.text(5,1.2,'Clustering HMP samples by Spearman correlation
distance',fontsize=35)
```



نلاحظ بوضوح 4 مجموعات تتوافق مع عينات الفم والأمعاء والجلد والمهبل من مشروع HMP. أكبر مجموعة هي عينات الفم،في حين أن ثاني أكبر تجمع يتوافق مع عينات Gut HMP.

إعداد البيانات وتدريب مصنف CNN

الآن نحن بصدد استخدام أسماء 48 جنسًا بكتيريًا محددًا خاصًا بالأنسجة لاستخراج معرّفات الجينوم المرجعية لجميع الأنواع والسلالات المقابلة من قاعدة بيانات RefSeq NCBI. يوضح المقتطف أدناه كيفية تنزيل واستخراج معرفات الجينوم المرجعية للأنواع والسلالات الخاصة بالأنسجة المقابلة للأجناس الـ 48 المختارة الخاصة بالأنسجة. لهذا الغرض، سنستخدم ملف مطابقة الاسم إلى المراجع MapBactName2ID.txt الذي يمكن تنزيله من github الخاص بي. تم إنشاء الملف عن طريق استخراج رؤوس الجينومات المرجعية ومطابقة أسماء ملفاتهم (GCF-ids) بالعناوين.

```
#Download bacterial reference genomes from NCBI RefSeq
wget
ftp://ftp.ncbi.nlm.nih.gov/genomes/refseq/bacteria/assembly_summary
.txt
grep 'Complete Genome' assembly_summary.txt \
> assembly_summary_complete_latest_reference_genomes.txt
awk -F "\t" '$12=="Complete Genome" && $11=="latest"{print $20}'
assembly_summary.txt \
> assembly_summary_complete_latest_reference_genomes_paths.txt

mkdir BacterialGenomes
for i in $(cat
assembly_summary_complete_latest_reference_genomes_paths.txt)
do
wget -P BacterialGenomes ${i}/*genomic.fna.gz
done

#Grep reference genome IDs by the selected genera names
grep -wFf tissue_specific_genera_names_oral_12genera.txt
MapBactName2ID.txt > \
tissue_specific_fasta_IDs_oral.txt
grep -wFf tissue_specific_genera_names_gut_12genera.txt
MapBactName2ID.txt > \
tissue_specific_fasta_IDs_gut.txt
grep -wFf tissue_specific_genera_names_skin_12genera.txt
MapBactName2ID.txt > \
tissue_specific_fasta_IDs_skin.txt
grep -wFf tissue_specific_genera_names_vagina_12genera.txt
MapBactName2ID.txt > \
tissue_specific_fasta_IDs_vagina.txt
```

الآن، عندما يكون لدينا معرّفات الجينوم المرجعي (ملفات fasta) المقابلة للأجناس الخاصة بالأنسجة، يمكننا تقسيم الجينومات المرجعية وإنشاء 4 مجموعات من التسلسلات (على سبيل المثال 80 nt طويلة) مع التسميات 0 و 1 و 2 و 3 لكل مجموعة تتوافق مع الأجناس الميكروبية للفم والأمعاء والجلد والمهبل.في الحلقة أدناه، قرأنا أولاً الملفات التي تحتوي على معرّفات fasta لكل من الأنسجة الأربعة،

ثم نقرأ ملفات fasta لكل من معرّفات fasta ونختار 3000 تسلسل عشوائي بطول 80 nt. أثناء ذلك، سنقوم أيضًا بإسقاط التسلسلات التي تحتوي على N و H و M وغيرها من النيوكليوتيدات غير المتعارف عليها ونقتصر على التسلسلات التي تحتوي على A و C و T و G فقط.

```
import gzip; import random; import pandas as pd

nt = 80 # length of each sequence
Nseq = 3000 # number of randomly drawn sequences from each fasta-
file
random.seed(123)

all_tissues_seqs = []
all_tissues_labs = []
for lab, tissue in enumerate(['oral', 'gut', 'skin', 'vagina']):

    print('Working with tissue: {}
***********************************'.format(tissue))
    fasta_df = pd.read_csv('tissue_specific_fasta_IDs_' + tissue +
'.txt',
    header = None, sep = '\t')

    tissue_seqs = []
    for i in range(fasta_df.shape[0]):

        with gzip.open(fasta_df[0][i], 'r') as f:
            content = f.readlines()
            del content[0]

            random_seqs = random.sample(content, Nseq)
            random_seqs = [j.rstrip()[0:nt].decode() for j in
random_seqs]
            random_seqs = [k for k in random_seqs if
                           k.count('A') > 0 and k.count('C') > 0
and k.count('G') > 0
                           and k.count('T') > 0 and len(list(k)) ==
nt and 'N' not in k
                           and 'H' not in k and 'K' not in k and
'M' not in k
                           and 'R' not in k and 'S' not in k and
'V' not in k
                           and 'W' not in k and 'Y' not in k and
'B' not in k]

            tissue_seqs = tissue_seqs + random_seqs

            if((i + 1) % 500 == 0):
                print('Finished {} fasta-files'.format(i + 1))

    all_tissues_seqs = all_tissues_seqs + tissue_seqs
    print(all_tissues_seqs[(len(all_tissues_seqs)-
3):len(all_tissues_seqs)])
    all_tissues_labs = all_tissues_labs + [lab]*len(tissue_seqs)
```

```
    print(all_tissues_labs[(len(all_tissues_labs)-
3):len(all_tissues_labs)])
```

بمجرد إنشاء مجموعة بيانات من التسلسلات ذات العلامات labelled sequences، حيث يحصل كل تسلسل على تسمية 0 أو 1 أو 2 أو 3 تتوافق مع Oral وGut وSkin وVagina، سيتعين علينا تحويل التسلسلات إلى موتر واحد مشفر ساخنًا one-hot-encoded tensor (مصفوفة numpy ثلاثية الابعاد) يمكن قراءته بواسطة خوارزمية التعلم العميق مثل 1D CNN. بعد ذلك، سنقوم أيضًا بتقسيم التسلسلات ذات التشفير الواحد الساخن إلى مجموعات فرعية للتدريب والاختبار.

```
import numpy as np
from sklearn.preprocessing import LabelEncoder, OneHotEncoder

input_features = []
integer_encoder = LabelEncoder(); one_hot_encoder = OneHotEncoder()
for sequence in all_tissues_seqs:
  integer_encoded = integer_encoder.fit_transform(list(sequence))
  integer_encoded = np.array(integer_encoded).reshape(-1, 1)
  one_hot_encoded = one_hot_encoder.fit_transform(integer_encoded)
  input_features.append(one_hot_encoded.toarray())
input_features = np.stack(input_features)

one_hot_encoder = OneHotEncoder()
labels = np.array(all_tissues_labs).reshape(-1, 1)
input_labels = one_hot_encoder.fit_transform(labels).toarray()

from sklearn.model_selection import train_test_split
train_features, test_features, train_labels, test_labels =
train_test_split(
    input_features, input_labels, test_size = 0.25, random_state =
42)
```

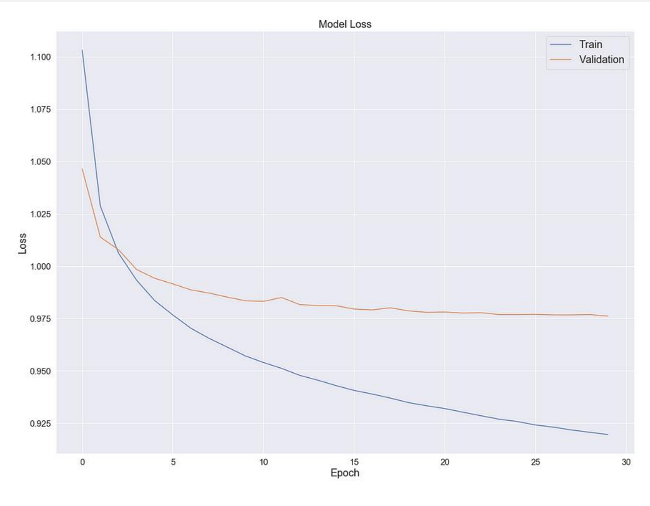
أخيرًا، دعنا نكتب بُنية بسيطة 1D CNN ونبدأفي تدريب المصنف على التعرف على أنماط التسلسل التي تتوافق مع 4 فئات هي المجتمعات الميكروبية عن طريق الفم والأمعاء والجلد والمهبل.

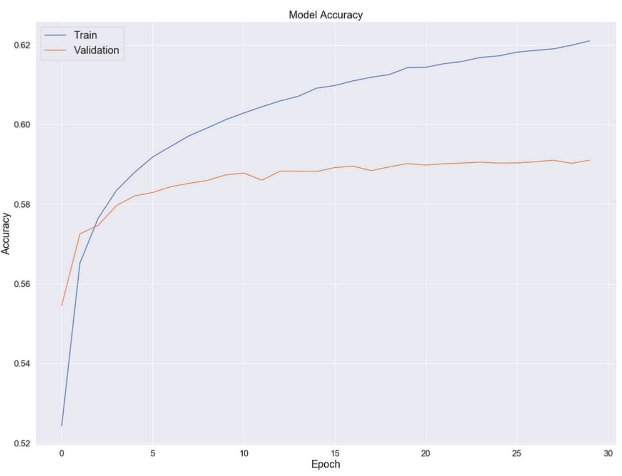
```
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.models import Sequential
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.callbacks import ReduceLROnPlateau
from tensorflow.keras.layers import
Conv1D,Dense,MaxPooling1D,Flatten,Dropout

model = Sequential()
model.add(Conv1D(filters=512,kernel_size=20,input_shape=(train_feat
ures.shape[1],4),
                    padding = 'same', activation = 'relu'))
model.add(MaxPooling1D(pool_size = 2))
model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dense(256, activation = 'relu'))
model.add(Dropout(0.1))
model.add(Dense(4, activation = 'softmax'))
```

```
epochs = 30; lrate = 0.01; decay = lrate / epochs
sgd = SGD(lr = lrate, momentum = 0.9, decay = decay, nesterov =
False)
model.compile(loss='categorical_crossentropy',optimizer=sgd,metrics
=['accuracy'])

checkpoint =
ModelCheckpoint("weights.best.hdf5",monitor='val_accuracy',verbose=
1,
                             save_best_only = True, mode = 'max')
rlrp = ReduceLROnPlateau(monitor = 'val_accuracy', factor = 0.1,
patience = 5,
                         verbose = 1, min_lr = 0.0001)
history = model.fit(train_features, train_labels, epochs = epochs,
verbose = 1,
                    validation_split = 0.25,batch_size = 32,
shuffle = True,
                    callbacks = [rlrp, checkpoint])
```





أفضل طريقة للتحقق مما إذا كانت الشبكة قد تعلمت تصنيف التسلسلات هي تقييم أدائها على مجموعة اختبار جديدة تتكون من بيانات لم تلاحظها على الإطلاق أثناء التدريب. هنا، نقوم بتقييم النموذج على مجموعة الاختبار ورسم النتائج كمصفوفة ارتباك confusion matrix.

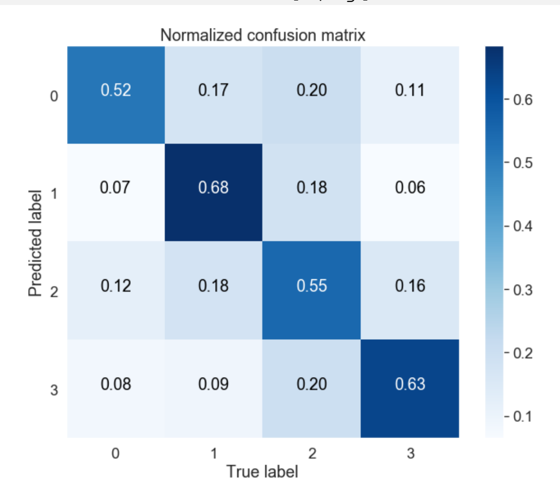
```
import itertools; import numpy as np
from sklearn.metrics import confusion_matrix
import seaborn as sns; import matplotlib.pyplot as plt

plt.figure(figsize = (10,8))

predicted_labels = model.predict(np.stack(test_features))
cm = confusion_matrix(np.argmax(test_labels, axis = 1),
                      np.argmax(predicted_labels, axis = 1))
print('Confusion matrix:\n',cm)

cm = cm.astype('float') / cm.sum(axis = 1)[:, np.newaxis]

plt.imshow(cm, cmap = plt.cm.Blues)
plt.title('Normalized confusion matrix')
plt.colorbar()
plt.xlabel('True label')
plt.ylabel('Predicted label')
plt.xticks([0, 1, 2, 3]); plt.yticks([0, 1, 2, 3])
plt.grid('off')
for i, j in itertools.product(range(cm.shape[0]),
range(cm.shape[1])):
    plt.text(j, i, format(cm[i, j], '.2f'),
             horizontalalignment = 'center',
             color = 'white' if cm[i, j] > 0.5 else 'black')
```




```
scores = model.evaluate(test_features, test_labels, verbose = 0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```

```
Accuracy: 59.52%
```

التمثيل الرسومي لمصفوفة الارتباك هو خريطة الحرارة الزرقاء blueish heatmap التي توضح بنية تشبه الكتلة block-like structure مما يعني أن المصنف 1D CNN كان قادرًا على التمييز بين الفئات الميكروبية الأربعة جيدًا. قد يبدو متوسط دقة التقييم الإجمالية البالغة 60٪ منخفضًا للوهلة الأولى لأن عقولنا مصممة على مشكلة تصنيف ثنائي حيث يمكن أن يعطي المصنف العشوائي دقة أساسية تبلغ 50٪. لذلك، قد تبدو نتيجتنا 60٪ أفضل قليلاً من المصنف العشوائي. ومع ذلك، يرجى تذكر أننا نتعامل هنا مع 4 فئات، وبالتالي يجب أن يمنحنا المصنف العشوائي دقة متوسطة تبلغ 25٪ (يمكن تأكيدها عن طريق المحاكاة، غير معروض هنا) مع انحراف معياري يبلغ حوالي 5-10٪ اعتمادًا على حجم العينة. لذلك، دقة 60٪ أعلى بكثير من القاعدة 25٪ التي يتوقعها المرء من مصنف عشوائي. علاوة على ذلك، سنستخدم المصنف المدرب أدناه للحصول على تقدير تقريبي لمساهمات المجتمع الميكروبيفي عينة ميتاجينومية عن طريق حساب متوسط التنبؤات من آلاف تسلسلات الحمض النووي. بافتراض أن دقة المصنف بنسبة 60٪ تسمح بالتنبؤات الصحيحة لغالبية التسلسلات، على الرغم من عدم تصنيفها جميعًا بشكل صحيح، إلا أن هذا لا يزال يمنحنا حدسًا جيداً حول المجتمع الميكروبي (الفم أو الأمعاء أو الجلد أو المهبل) الذي جاءت منه العينة.

عمل تنبؤات عن التركيب الجرثومي

نحن هنا بصدد التحقق مما إذا كان يمكن استخدام مصنف CNN المدرب لاستنتاج التركيب الميكروبي microbial composition لعينة ميتاجينومية عشوائية. لهذا الغرض، سنختار ملفات metagenomic fastq العشوائية (من قواعد بيانات HMP أو MGnify أو DIABIMMUNE)، ونضع تنبؤات لكل تسلسل ميتاجينومي، ونعرض كسور التسلسلات التي تم تخصيصها لكل فئة من الفئات الأربعة (الفم، والأمعاء، والجلد والمهبل)، التي تتوافق تقريبًا مع أجزاء المجتمعات الميكروبية (التركيب الميكروبي microbial composition) الموجودةفي كل عينة. لنبدأ ببعض العينات من مشروع HMP. أعلم أننا استخدمنا وفرة HMP لاختيار العلامات الميكروبية الخاصة بالأنسجة (لم نستخدم بيانات التسلسل الخام raw sequencing data)، ويمكن أن يكون هناك تحيزفي استخدام ملفات HMP fastq لاستنتاج تركيبتها الميكروبية. ومع ذلك، فهذه بداية جيدة، على الأقل إذا فشل مصنف CNNفي عينات HMP، فمن الصعب توقع أداء جيدفي العينات المستقلة. لاحقًا، سوف نكرر هذا الاستنتاج باستخدام عينات ميتاجينومية مستقلة حقًا من مشاريع أخرى غير مشاريع HMP.في الحلقة أدناه، قرأت ملفات HMP fastq (اخترت عشوائيًا عينتين من مسحات الفم، والأمعاء (البراز Stool)، والجلد والمهبل)، وقمت بترميز تسلسلاتهم الساخنة وتشغيل التنبؤات باستخدام مصنف CNN المدرب. بعد ذلك، قمت بتحديد أكثر التنبؤات موثوقية حيث حصل مجتمع ميكروبي واحد على احتمال 80٪ على الأقل. أخيرًا، أرسم كسور fractions التسلسلات التي حددها المصنف لكل مجتمع ميكروبي.

```
import numpy as np
```

```
from Bio import SeqIO
from tensorflow import keras
from collections import Counter
from sklearn.preprocessing import LabelEncoder, OneHotEncoder

tissue_list = ['Oral', 'Oral', 'Gut', 'Gut', 'Skin', 'Skin',
'Vagina', 'Vagina']
sample_list = ['SRS011243', 'SRS055495', 'SRS017247', 'SRS022713',
'SRS017156',
               'SRS018369', 'SRS062752', 'SRS015071']

nt = 80; cutoff = 0.8
oral_list = []; gut_list = []; skin_list = []; vagina_list = []
for j in range(len(sample_list)):

    test_seq_list = []
    fastq_file = 'HMP_' + tissue_list[j] + '_' + sample_list[j] + \
    '.denovo_duplicates_marked.trimmed.singleton.fastq'
    for record in SeqIO.parse(fastq_file, "fastq"):
        if len(str(record.seq)) >= 80:
            test_seq_list.append(str(record.seq))

    test_seq_list = [j.rstrip()[0:nt] for j in test_seq_list]
    test_seq_list = [k for k in test_seq_list if
                     k.count('A') > 0 and k.count('C') > 0 and
k.count('G') > 0
                     and k.count('T') > 0 and len(list(k)) == nt
and 'N' not in k
                     and 'H' not in k and 'K' not in k and 'M' not
in k
                     and 'R' not in k and 'S' not in k and 'V' not
in k
                     and 'W' not in k and 'Y' not in k and 'B' not
in k]

    input_test_features = []
    integer_encoder = LabelEncoder(); one_hot_encoder =
OneHotEncoder()
    for sequence in test_seq_list:
      integer_encoded =
integer_encoder.fit_transform(list(sequence))
      integer_encoded = np.array(integer_encoded).reshape(-1, 1)
      one_hot_encoded =
one_hot_encoder.fit_transform(integer_encoded)
      input_test_features.append(one_hot_encoded.toarray())
    input_test_features = np.stack(input_test_features)

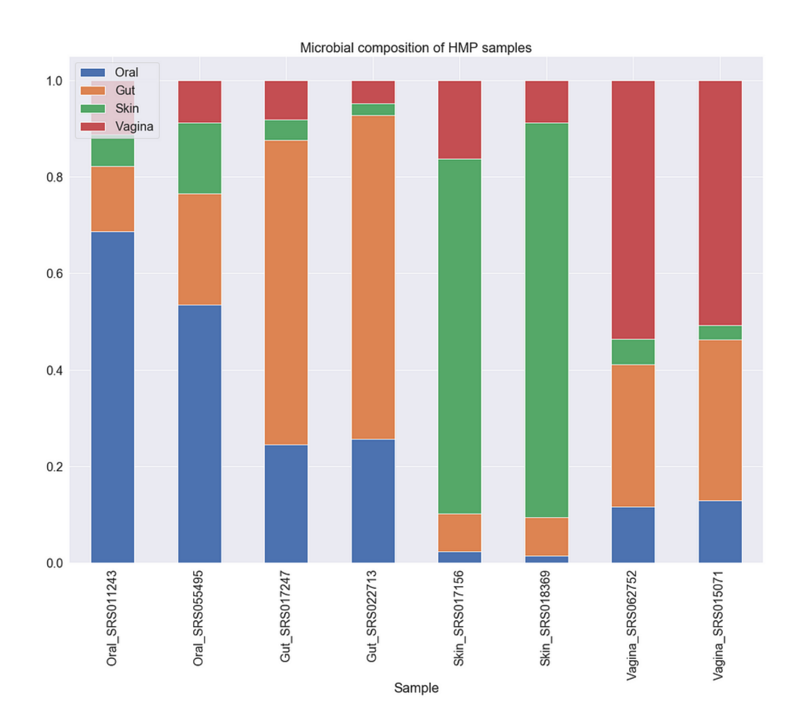
    predicted_test_probs =
model.predict(np.stack(input_test_features))
    microbial_fractions=\
    [Counter([np.argmax(j) for j in [i for i in
predicted_test_probs
                                     if max(list(i)) >
cutoff]])[0],
```

```
     Counter([np.argmax(j) for j in [i for i in
predicted_test_probs
                                             if max(list(i)) >
cutoff]])[1],
     Counter([np.argmax(j) for j in [i for i in
predicted_test_probs
                                             if max(list(i)) >
cutoff]])[2],
     Counter([np.argmax(j) for j in [i for i in
predicted_test_probs
                                             if max(list(i)) >
cutoff]])[3]]
    microbial_fractions = [i/sum(microbial_fractions) for i in
microbial_fractions]

    oral_list.append(microbial_fractions[0]);
gut_list.append(microbial_fractions[1])
    skin_list.append(microbial_fractions[2]);
vagina_list.append(microbial_fractions[3])

df = pd.DataFrame({'Sample': [i + '_' + j for i, j in
zip(tissue_list, sample_list)],
                   'Oral': oral_list, 'Gut': gut_list, 'Skin':
skin_list,
                   'Vagina': vagina_list})
df.plot(x = 'Sample', kind = 'bar', stacked = True,
        title = 'Microbial composition of HMP samples', figsize =
(20,15))
plt.legend(loc = 'upper left', prop = {'size': 20})
```

	Sample	Oral	Gut	Skin	Vagina
0	Oral_SRS011243	0.686508	0.135582	0.065146	0.112765
1	Oral_SRS055495	0.535085	0.229923	0.147645	0.087347
2	Gut_SRS017247	0.244642	0.630946	0.042865	0.081547
3	Gut_SRS022713	0.256093	0.672017	0.024034	0.047856
4	Skin_SRS017156	0.023256	0.079050	0.734708	0.162986
5	Skin_SRS018369	0.014902	0.080032	0.817496	0.087570
6	Vagina_SRS062752	0.116100	0.295430	0.052072	0.536397
7	Vagina_SRS015071	0.129196	0.333672	0.028993	0.508138



يمكننا أن نرى أننا نحصل على تنبؤات جيدة إلى حد ما لكل عينة عشوائية من HMP، أي أن أول اثنين يشبهان العينات عن طريق الفم لأن الميكروبات الفموية (الشريط الأزرق) تشكل معظم المجتمع الميكروبي،في حين أن النوعين الأخيرين يشبهان عينات الأمعاء وما إلى ذلك. تشير التسميات "Oral_" و "Gut_" وما إلى ذلك لكل عينةفي الجزء السفلي من الشكل أعلاه إلى الأنسجة الحقيقية حيث تم جمع العيناتفي مشروع HMP.

باختصار، قمنا بتدريب مصنف CNN باستخدام تسلسل الجينوم المرجعي للميكروبات الوفيرةفي الأنسجة المختلفة من مشروع HMP. بعد ذلك، شرعنافي تقييم النموذج على عدد قليل من عينات HMP العشوائية. ربما يمكنك رؤية مشكلة محتملة هنا. من ناحية أخرى، لم نستخدم تسلسلات أولية من عينات HMP عند تدريب مصنف CNN، وبدلاً من ذلك استخدمنا الجينومات المرجعية للأجناس الأكثر وفرةفي بيانات HMP. من ناحية أخرى، لا يزال هناك تسرب للمعلومات information leakage لأننا ما زلنا نستخدم نفس المشروع (على الرغم من وجود أنواع مختلفة من البيانات) للتدريب والاختبار. لذلك قد يظل التقييم متحيزًا ويبدو جيداً لدرجة يصعب تصديقها. لإجراء تقييم أفضل، سنختار 8 عينات عشوائية عن طريق الفم، والأمعاء، والجلد، والمهبل من المشاريع الميتاجينومية التي لا تتعلق بـ HMP.

```
From collections import Counter
import random; import numpy as np
from Bio import SeqIO; from tensorflow import keras
```

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder

random.seed(123)

tissue_list = ['Oral', 'Oral', 'Gut', 'Gut', 'Skin', 'Skin',
'Vagina', 'Vagina']
sample_list = ['ERR3827177.fastq', 'ERR3827196.fastq',
'G69146_pe_1.fastq',
               'G69153_pe_1.fastq', '4491200149-
1L1_S12_L001_R1_001.fastq',
               '4491200209-1L1_S66_L001_R1_001.fastq',
'SRR6900017_1.fastq',
               'SRR6899929_1.fastq']

nt = 80; cutoff = 0.9
oral_list = []; gut_list = []; skin_list = []; vagina_list = []
for j in range(len(sample_list)):

    test_seq_list = []
    fastq_file = sample_list[j]
    for record in SeqIO.parse(fastq_file, "fastq"):
        if len(str(record.seq)) >= nt:
            test_seq_list.append(str(record.seq))

    test_seq_list = random.sample(test_seq_list, 100000)
    test_seq_list = [j.rstrip()[0:nt] for j in test_seq_list]
    test_seq_list = [k for k in test_seq_list if
                     k.count('A') > 0 and k.count('C') > 0 and
k.count('G') > 0
                     and k.count('T') > 0 and len(list(k)) == nt
and 'N' not in k
                     and 'H' not in k and 'K' not in k and 'M' not
in k
                     and 'R' not in k and 'S' not in k and 'V' not
in k
                     and 'W' not in k and 'Y' not in k and 'B' not
in k]

    integer_encoder = LabelEncoder()
    one_hot_encoder = OneHotEncoder()
    input_test_features = []
    for sequence in test_seq_list:
      integer_encoded =
integer_encoder.fit_transform(list(sequence))
      integer_encoded = np.array(integer_encoded).reshape(-1, 1)
      one_hot_encoded =
one_hot_encoder.fit_transform(integer_encoded)
      input_test_features.append(one_hot_encoded.toarray())
    input_test_features = np.stack(input_test_features)

    predicted_test_probs =
model.predict(np.stack(input_test_features))
    if sum(Counter([np.argmax(j) for j in [i for i in
predicted_test_probs \
```

```
                                                    if
max(list(i))>cutoff]]).values())<500:
        cutoff = 0.6
    microbial_fractions=\
    [Counter([np.argmax(j) for j in [i for i in
predicted_test_probs
                                     if max(list(i)) >
cutoff]])[0],
     Counter([np.argmax(j) for j in [i for i in
predicted_test_probs
                                     if max(list(i)) >
cutoff]])[1],
     Counter([np.argmax(j) for j in [i for i in
predicted_test_probs
                                     if max(list(i)) >
cutoff]])[2],
     Counter([np.argmax(j) for j in [i for i in
predicted_test_probs
                                     if max(list(i)) >
cutoff]])[3]]

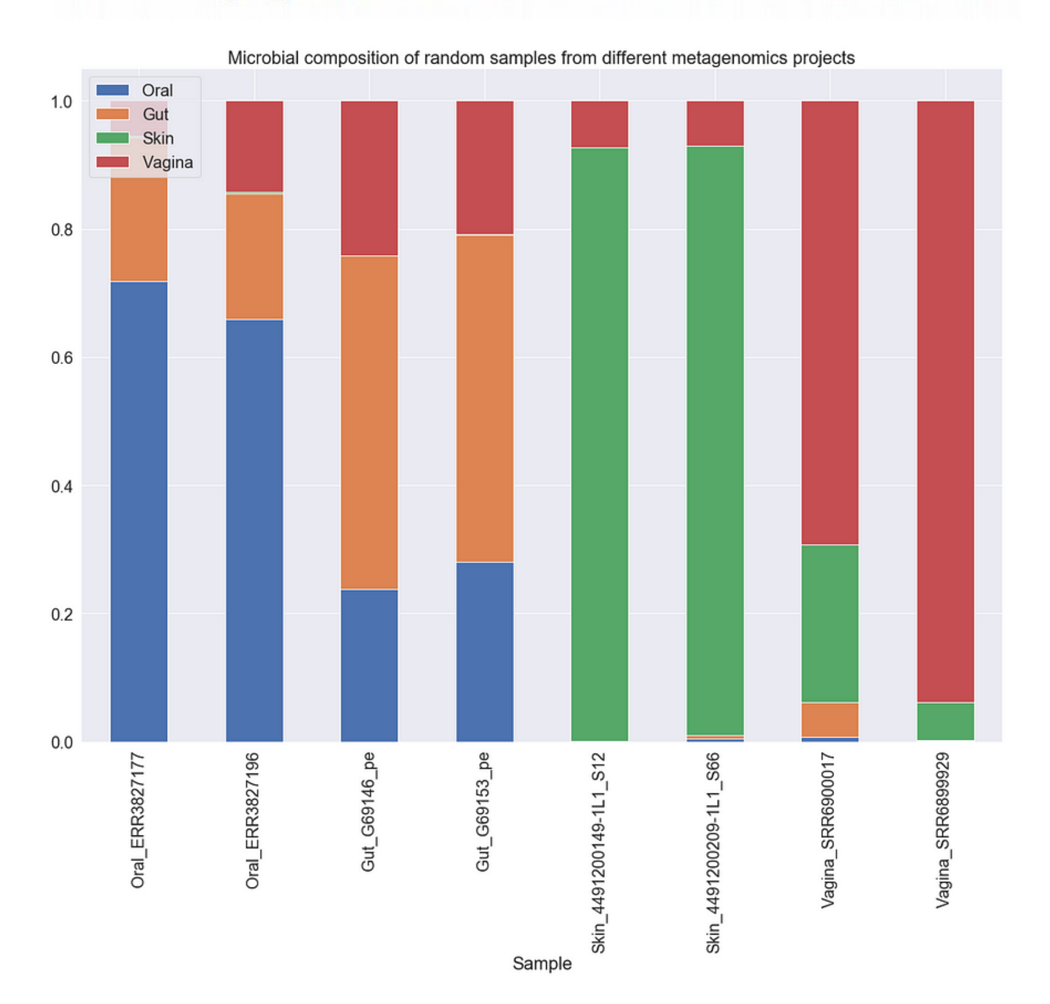
    microbial_fractions = [i / sum(microbial_fractions) for i in
microbial_fractions]
    print('Probability cutoff: {}'.format(cutoff))
    print(microbial_fractions)

    oral_list.append(microbial_fractions[0])
    gut_list.append(microbial_fractions[1])
    skin_list.append(microbial_fractions[2])
    vagina_list.append(microbial_fractions[3])

import pandas as pd
df = pd.DataFrame({'Sample': [i +'_'+j for i,j in zip(tissue_list,

[k.replace('.fastq','').replace('_1','').replace('_001','').replace
('_R1','')
          for k in sample_list])], 'Oral': oral_list, 'Gut':
gut_list,
                   'Skin': skin_list, 'Vagina': vagina_list})
df.plot(x = 'Sample', kind = 'bar', stacked = True,
        title='Microbial composition of other than HMP metagenomic
projects',
        figsize = (20,15))
plt.legend(loc = 'upper left', prop = {'size': 20})
```

	Sample	Oral	Gut	Skin	Vagina
0	Oral_ERR3827177	0.717918	0.224576	0.003027	0.054479
1	Oral_ERR3827196	0.659031	0.195681	0.002618	0.142670
2	Gut_G69146_pe	0.238034	0.520740	0.000000	0.241225
3	Gut_G69153_pe	0.280303	0.510101	0.001684	0.207912
4	Skin_4491200149-1L1_S12	0.000699	0.000806	0.924974	0.073521
5	Skin_4491200209-1L1_S66	0.004673	0.005340	0.919893	0.070093
6	Vagina_SRR6900017	0.007692	0.053846	0.246154	0.692308
7	Vagina_SRR6899929	0.000569	0.001138	0.059727	0.938567



هنا نرى مرة أخرى أن تنبؤات التراكيب الميكروبيةفي 8 عينات من غير HMP تتفق إلى حد ما مع المجتمعات الميكروبية الحقيقية للفم، والأمعاء، والجلد، والمهبل التي تم أخذ العينات منها. أحسنت! يمكن لمصنف CNN المدرب لدينا تقديم تقديرات ذات مغزى للتكوين الميكروبي لأي عينة معينة.

الاستنتاج

في هذه المقالة، تعلمنا أن الميتاجينومية Metagenomics تمثل بيانات ضخمة مناسبة لتحليل التعلم الآلي والعميق. استخدمنا بيانات metagenomics من مشروع Human Microbiome (HMP)، وطورنا مصنفاً 1D CNN يمكنه التنبؤ بنجاح بالمساهمات من المجتمعات الميكروبية المختلفة (الفم أو الأمعاء أو الجلد أو المهبل) لعينة معينة بدءًا من التسلسلات الأوليةفي بيانات fastq القياسية الشكل دون إجراء تقدير كمي للوفرة الميكروبية. ربما تقدم هذه الطريقة مزايا مقارنة باستدلال التركيب الميكروبي القياسي باستخدام SourceTracker.

المصدر

https://towardsdatascience.com/deep-learning-on-human-microbiome-7854fba815fc

16) تصنيف تسلسل DNA للتنبؤ بالأنواع DNA Sequence Classification for Species Prediction

يوجد الحمض النووي DNA في خلايا كل كائن حي على الأرض. يمكن للعلماء دراسة الحمض النووي للكائنات الحية لفهم المزيد حول وظيفة الكائن الحي أو تحليل الحمض النووي لتحديد ما إذا كانوا قد وجدوا نوعًا جديدًا.

تصنيف الحمض النووي له العديد من الاستخدامات – من تحديد الأنواع species identification إلى التمايز differentiation، يمكن أن يوفر البحث رؤى حول كيفية اختلاف نوع واحد عن الآخر، أو كيفية تطور نوع واحد. يمكن لتحليل التسلسل الجيني Genetic sequence analysis تحديد الاختلافات في الجينات، وعمليات الإدخال، والحذف داخل مجموعات فرعية من الحمض النووي، أو تحديد التغيير في الأنماط الظاهرية من الأنواع إلى الأنواع. يمكن أن يخبرنا تحليل الاختلافات في التسلسلات من الأنواع المختلفة كثيرًا عن الاختلافات بين الأنواع، ولكن أيضًا عن مدى تشابه جميع الكائنات الحية.

من خلال تحسين النمذجة في تحليل تسلسل الحمض النووي، يمكن للباحثين الجينوم تحقيق نتائج أكثر دقة وأسرع لتصنيف وتمييز الأنواع من عينات الحمض النووي.

بالنسبة لهذا المشروع، سيكون تركيزي الأساسي على إظهار نماذج وطرق التصنيف. لاختيار مثال، قررت البحث في تحليل تسلسل الحمض النووي، لأنني اعتقدت أنه سيكون مثيرًا للاهتمام، والبيانات المتاحة للجمهور هائلة بشكل مذهل.

الفرضية

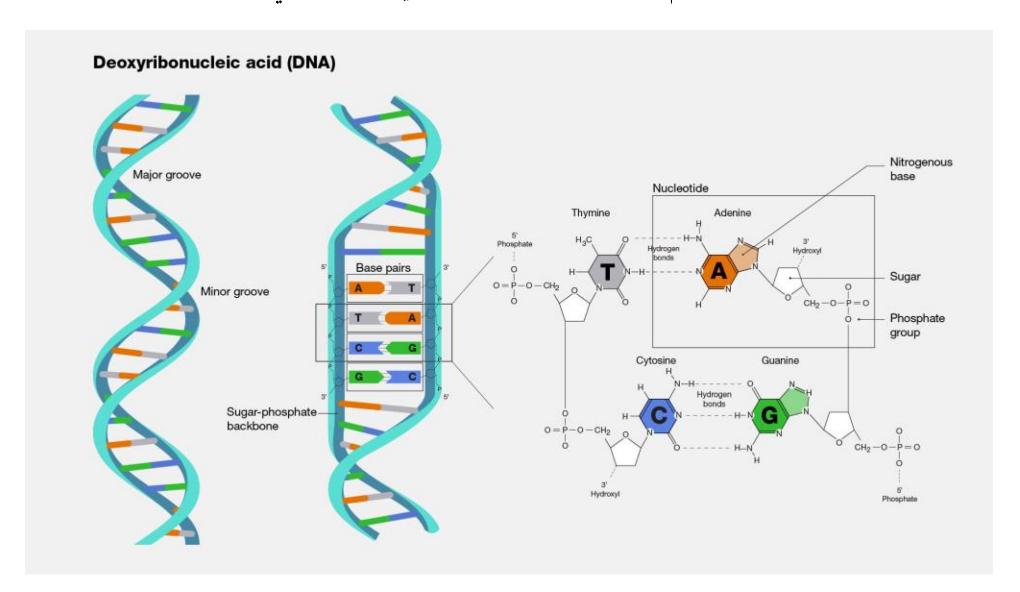
ماذا لو كان بإمكاني أخذ معلومات الحمض النووي من جينوم genome معين ومقارنتها بجينومات أخرى؟ هل ستكون هناك إشارة في الضوضاء؟ بالطبع، سيكون هناك وسيوجد، لكنني بدأت بفرضية العدم في السؤال: هل يمكن أن تكون هذه الجينومات غير مرتبطة أو مجرد تسلسلات عشوائية عند مقارنتها؟

بدلاً من ذلك، آمل أن أجد نمط التشابه، وآمل أن يتم قياس ذلك في مقاييس نموذجي. أنا لست عالم أحياء، لكن كعالم بيانات data scientist، سأنظر إلى هذه المشكلة على أنها مشكلة ملاحظة خطأ loss observing ــــ التشابه بين الجينومات وشرح التباين بينهما من خلال التنبؤ بفئة كل منهما. من وجهة نظر فنية، سيستخدم هذا المشروع طرق التعلم الآلي لتحليل وتصنيف تسلسل الحمض النووي من الأنواع المختلفة. سيتم استخدام النموذج الناتج للتنبؤ بأنواع مجموعة معينة من الفئات بناءً على التسلسل الجيني genetic sequence وحده. سيركز هذا المشروع في المقام الأول على الاختلافات الرياضية بين التسلسلات في الكود الجيني genetic code، بدلاً من أي تحليل جيني خاص للجينات

والأنماط الظاهرية بين الأنواع. أنا مهتم بمشكلة كيف يمكن أن يكون نوعان متشابهين أو مختلفين من نوع آخر، من وجهة نظر معممة وإحصائية.

ما هو DNA؟

الحمض النووي DNA، أو الحمض النووي الريبي منقوص الأكسجين deoxyribonucleic acid، هو جزيء يحتوي على الشفرة الموروثة inherited code التي تستخدمها الخلايافي الكائن الحي لتوليد البروتينات. يوجدفي نواة الخلايا. يتكون الحمض النووي من أربع قواعد: الأدينين adenine أو A ، أو السيتوزين cytosine أو C ، أو الجوانين guanine أو G ، أو الثايمين thymine أو T. كل خيط strand من الحمض النووي يتكون من خيطين متماسكين معًا بواسطة بنية أساسها السكر على شكل حلزون مزدوج. كل قاعدة متصلة بالخيط المعاكس من خلال قاعدتها التكميلية: يتصل A دائمًا بـ T، ويتصل C دائمًا بـ G، ويتصل G دائمًا بـ C، ويتصل T دائمًا بـ A. وهذا يعني أن كل خيط مكمل من الحمض النووي يشبه الانعكاس من جهة أخرى. عندما يتم تسلسل الحمض النوويفي سلسلة مكونة من القواعد الأربعة، فقد تم فهرسة سلسلة واحدة فقطفي البيانات التي أقرأها، بدون مكملها.



من خلال العمليات البيولوجية والجزيئية المعقدة، يكرر الحمض النووي أو يخلق نسخًا من نفسه تُعرف باسم mRNA. خيوط mRNA هي عبارة عن نسخ مجزأة قصيرة من الحمض النووي DNA يمكن أن تتحد مع أجزاء أخرى من mRNA، وبالتالي تخلق مجموعات تكون رمزًا لإنشاء البروتينات. تخلق التركيبات المختلفة بروتينات مختلفة ثم تُستخدم هذه البروتيناتفي جميع أنحاء الخلية والكائن الحي. يعمل الحمض النووي مثل كود الكمبيوتر لإنشاء البروتينات مثل البرنامج، وبالتالي لماذا يشار إليه على

أنه رمز جيني أو اللبنات الأساسية للحياة. يمكن تحديد مجموعات فرعية من خيوط DNA لإنشاء بروتينات معينة وتعرف باسم الجينات genes.

يتم لف الحمض النوويفي كل خلية معًا بإحكام، وعلى الرغم من أنه يشغل مساحة ميكروسكوبية، إلا أنه طويل جدًا. إذا قمت بتفكيك خيط كامل من الحمض النووي البشري بعناية، فسيكون طوله حوالي 1.8 مترًا ويحتوي على 3 مليارات زوج قاعدي، أي مجموعات من A و C و G و T.

سمح التقدم التكنولوجي للعلماء بتسلسل البيانات المجهرية ضمن هامش خطأ صغير. يُعرف العدد الإجمالي للجيناتفي أحد الأنواع بجينوم هذا النوع species' genome. قام مشروع الجينوم البشري The Human Genome Project، الذي استمر من عام 1990 حتى اكتمالهفي عام 2003، بتعيين الجينوم البشري بأكمله عبر حوالي 20000 عينة غير متداخلة. اليوم، هناك العديد من جينومات الأنواع المختلفة المعينة والمتاحة للجمهور للتنزيل للبحث والتحليل.

طرق هندسة خصائص تسلسل الحمض النووي الشائعة

علم الجينوم Genomics هو مجال واسع للدراسة يصنف ويحلل ويرسم خرائط جينومات الكائنات الحية. ضمن علم البيانات الجينومية genomic data science، هناك العديد من الطرق التي يمكن من خلالها استخدام التعلم الآلي لتحليل بيانات التسلسل.

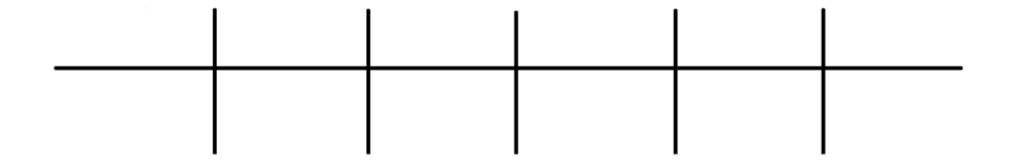
إذا تم أخذ خيط من الحمض النووي كمدخل، فقد تظهر السلسلة كمجموعة من الأحرف A و C و G و T ، مثل: "ATGAAGAAGGTAACTGCAGAGGCTATTTCCTGGAAT ..."

تسلسلات الحمض النووي هي مجموعات من A و C و T و G

محول K-Mer

طريقة أخرى شائعة للاستخدام، خاصة إذا كان طول التسلسل متغيرًا، تسمى .minhash Minhashing هي طريقةفي علوم الكمبيوتر لمقارنة التشابه بين مجموعتين.في المعلوماتية الحيوية bioinformatics، يمكن استخدام minhashing لتحويل تسلسل إلى قاموس متكرر لسلاسل فرعية بطول k تسمى k-mers لعدد صحيح معين k. أولاً، يتم إنشاء الأعمدة لجميع k-mers الممكنة لـ A و C و G و T . ثم يتم ملء كل عمود تبديل لكل مجموعة فرعية بحجم k من السلسلة. على سبيل المثال، يمكن تحويل سلسلة "gtatca" إلى k-mers لـ 2 = k عن طريق استخراج كل السلاسل الفرعية ذات الطول 2 على طول السلسلة. من أعمدة "gt" — "gtatca" و "ta" و "at" و "tc" و "ca" سيكون لكل منها قيمة 1.

يوجد أدناه رسم متحرك يوضح هذه الطريقة عندما يكون 3 = k.



البيانات

لقد استخدمت خمسة جينومات مختلفة من قاعدة بيانات RefSeq: بيانات الحمض النووي للانسان Human والشمبانزي Chimpanzee والدولفين Dolphin وشجرة البلوط Oak Tree والفطر Mushroom. كنت أرغبفي اختيار عدد قليل من الجينومات التي قد تكون متشابهة للغاية (بشري، شمبانزي، دولفين)، بالإضافة إلى زوجين سيكونان أقل قليلاً.

نوبتبوك جوبيتر

لقد قمت بترميز هذا المشروع بأكملهفي Pythonفي Jupyter Notebook والذي يمكن العثور عليه هنا.

فيما يلي ملخص لعمليتي التكرارية:

استيراد تسلسل DNA

سأستخدم مكتبة BioPython لتحليل ملفات FASTA الموجودةفي قاعدة بيانات RefSeq.

يحتوي كل ملف على تسلسلات من جينوم الأنواع. لنلق نظرة على سجل واحد كمثال على البيانات التي يتم تحليلها.

```
From Bio import SeqIOfor seq_record in
SeqIO.parse("data/GCF_000001405.40_GRCh38.p14_cds_from_genomic.fna",
"fasta"): #human fasta file
    print(seq_record)
    break #break after printing one recordID:
lcl|NC_000001.11_cds_NP_001005484.2_1
Name: lcl|NC_000001.11_cds_NP_001005484.2_1
Description: lcl|NC_000001.11_cds_NP_001005484.2_1 [gene=OR4F5]
[db_xref=CCDS:CCDS30547.1,Ensembl:ENSP00000493376.2,GeneID:79501]
[protein=olfactory receptor 4F5] [protein_id=NP_001005484.2]
[location=join(65565..65573,69037..70008)] [gbkey=CDS]
Number of features: 0
Seq('ATGAAGAAGGTAACTGCAGAGGCTATTTCCTGGAATGAATCAACGAGTGAAACG...TAG')
```

سيستخدم هذا المشروع التسلسل (Seq) كبياناتفي النموذج، لكنني سأحدد دالة مساعدة ستجمع عددًا قليلاً من أعمدة البيانات، فقطفي حالة رغبتفي الرجوع إليها لاحقًا.

	seq	gene	protein	target
0	ATGAAGAAGGTAACTGCAGAGGCTATTTCCTGGAATGAATCAACGA...	OR4F5	olfactory receptor 4F5	human
1	ATGCCTAGACACACACATCCTTACTCTGCGTGCATCCCTGGCCTGG...	LOC112268260	uncharacterized protein LOC112268260	human
2	ATGGATGGAGAGAATCACTCAGTGGTATCTGAGTTTTTGTTTCTGG...	OR4F29	olfactory receptor 4F3/4F16/4F29	human
3	ATGCGTAGACACACACATCCTTACTCTGCGCGCATCCCTGGCCTGG...	LOC105378947	proline-rich extensin-like protein EPR1 isofor...	human
4	ATGGATGGAGAGAATCACTCAGTGGTATCTGAGTTTTTGTTTCTGG...	OR4F16	olfactory receptor 4F3/4F16/4F29 isoform X1	human

دعونا نلقي نظرة على تسلسل الحمض النووي الأول بطوله الإجمالي.

```
Human_df.loc[0,'seq']'ATGAAGAAGGTAACTGCAGAGGCTATTTCCTGGAATGAATCAACGAGT
GAAACGAATAACTCTATGGTGACTGAATTCATTTTTCTGGGTCTCTCTGATTCTCAGGAACTCCAGACCT
TCCTATTTATGTTGTTTTTTGTATTCTATGGAGGAATCGTGTTTGGAAACCTTCTTATTGTCATAACAGT
GGTATCTGACTCCCACCTTCACTCTCCCATGTACTTCCTGCTAGCCAACCTCTCACTCATTGATCTGTCT
CTGTCTTCAGTCACAGCCCCCAAGATGATTACTGACTTTTTCAGCCAGCGCAAAGTCATCTCTTTCAAGG
GCTGCCTTGTTCAGATATTTCTCCTTCACTTCTTTGGTGGGAGTGAGATGGTGATCCTCATAGCCATGGG
CTTTGACAGATATATAGCAATATGCAAGCCCCTACACTACACTACAATTATGTGTGGCAACGCATGTGTC
GGCATTATGGCTGTCACATGGGGAATTGGCTTTCTCCATTCGGTGAGCCAGTTGGCGTTTGCCGTGCACT
TACTCTTCTGTGGTCCCAATGAGGTCGATAGTTTTTATTGTGACCTTCCTAGGGTAATCAAACTTGCCTG
TACAGATACCTACAGGCTAGATATTATGGTCATTGCTAACAGTGGTGTGCTCACTGTGTGTTCTTTTGTT
CTTCTAATCATCTCATACACTATCATCCTAATGACCATCCAGCATCGCCCTTTAGATAAGTCGTCCAAAG
CTCTGTCCACTTTGACTGCTCACATTACAGTAGTTCTTTTGTTCTTTGGACCATGTGTCTTTATTTATGC
CTGGCCATTCCCCATCAAGTCATTAGATAAATTCCTTGCTGTATTTTATTCTGTGATCACCCCTCTCTTG
AACCCAATTATATACACACTGAGGAACAAAGACATGAAGACGGCAATAAGACAGCTGAGAAAATGGGATG
CACATTCTAGTGTAAAGTTTTAG'
```

أريد إنشاء جميع الميزات في نموذجي من تسلسلات كهذه. لقد كتبت فئة TransformerMixin مخصصة هنا لتحويل العمود "seq" إلى قواميس تكرار k-mer التي تمت مناقشتها أعلاه ثم تحويلها إلى إطار بيانات Pandas. تسمى هذه الدالة KmerTransformer ويمكن العثور عليها على github الخاص بي.

تحليل البيانات الاستكشافية

قبل أن أقوم بأي تحليل، سأضيف جميع الأنواع من بيانات تسلسل الحمض النووي التي جمعتها في إطار بيانات واحد.

	seq	gene	protein	target
0	ATGAAGAAGGTAACTGCAGAGGCTATTTCCTGGAATGAATCAACGA...	OR4F5	olfactory receptor 4F5	human
1	ATGCCTAGACACACACATCCTTACTCTGCGTGCATCCCTGGCCTGG...	LOC112268260	uncharacterized protein LOC112268260	human
2	ATGGATGGAGAGAATCACTCAGTGGTATCTGAGTTTTTGTTTCTGG...	OR4F29	olfactory receptor 4F3/4F16/4F29	human
3	ATGCGTAGACACACACATCCTTACTCTGCGCGCATCCCTGGCCTGG...	LOC105378947	proline-rich extensin-like protein EPR1 isofor...	human
4	ATGGATGGAGAGAATCACTCAGTGGTATCTGAGTTTTTGTTTCTGG...	OR4F16	olfactory receptor 4F3/4F16/4F29 isoform X1	human

أول شيء نريد رؤيته هو مقاييس تسلسل الحمض النووي نفسه. دعونا نلقي نظرة على طول التسلسل، مجمعة حسب الأنواع.

target	chimp	dolphin	human	mushroom	oak
count	20000.000000	20000.000000	20000.000000	13330.000000	20000.000000
mean	2517.103150	2053.371500	2223.870600	1459.588297	1516.557900
std	5673.450049	1784.546128	3788.036344	1114.727843	1368.491338
min	108.000000	108.000000	37.000000	153.000000	93.000000
25%	996.000000	969.000000	939.000000	729.000000	753.000000
50%	1612.500000	1554.000000	1515.000000	1167.000000	1209.000000
75%	2763.000000	2568.000000	2592.000000	1806.000000	1875.000000
max	106962.000000	28215.000000	107976.000000	13854.000000	23319.000000

يمكننا الآن تحديد أن متوسط طول كل تسلسل بين الأهداف هو نفسه نسبيًا، بصرف النظر عن البيانات الجينومية لشجرة البلوط والفطر.

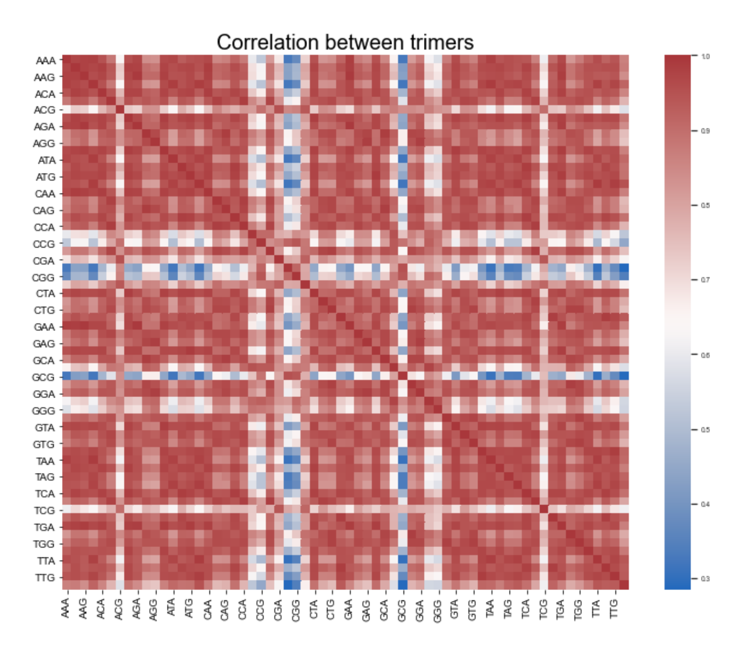
باستخدام دالة KmerTransformer المخصصة التي قمت بإنشائها، يمكنني تحويل البيانات وإجراء بعض التحليلات المرئية الأساسية. سأبدأ بـ k = 3 لذلك ليس لدي الكثير من الأعمدة ويمكنني رؤية البيانات معًا بسهولة.

دعنا نفحص بسرعة أن البيانات يتم ملؤها في الأعمدة بشكل صحيح.

	AAA	AAC	AAG	AAT	ACA	ACC	ACG	ACT	AGA	AGC	...	TCT	TGA	TGC	TGG	TGT	TTA	TTC	TTG	TTT	target
0	10	13	13	16	19	10	4	21	15	11	...	34	23	13	21	29	16	29	20	35	human
1	9	20	14	4	30	18	14	13	41	19	...	25	10	45	44	12	5	7	9	9	human
2	7	5	12	10	17	10	3	22	15	14	...	31	14	10	30	33	11	33	17	33	human
3	8	30	15	4	44	34	13	15	64	30	...	52	9	84	70	14	4	10	6	7	human
4	7	5	12	10	17	10	3	22	15	14	...	31	14	10	30	33	11	33	17	33	human

5 rows × 65 columns

وإلقاء نظرة على الارتباط بين الأعمدة ...



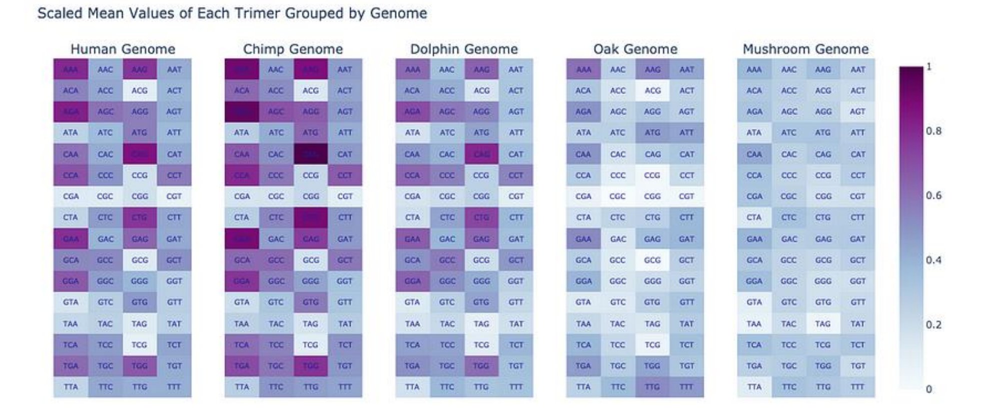
توضح خريطة الحرارة heatmap هذه أن هناك ارتباطًا كبيرًا بين معظم ميزاتنا، وهذا ليس مفاجئًا لأن كل k-mer له سلاسل فرعية مشتركة مع أعمدة k-mer الأخرى. سيؤثر هذا على نموذجنا، وقد أتمكن من إيجاد طريقة لاستخراج الميزات التي لا ترتبط أو تستخدم المصنفات التي يمكنها التعامل مع مستوى عالٍ من العلاقة الخطية.

متوسط القيم لكل مقلم مجمعة حسب الأنواع

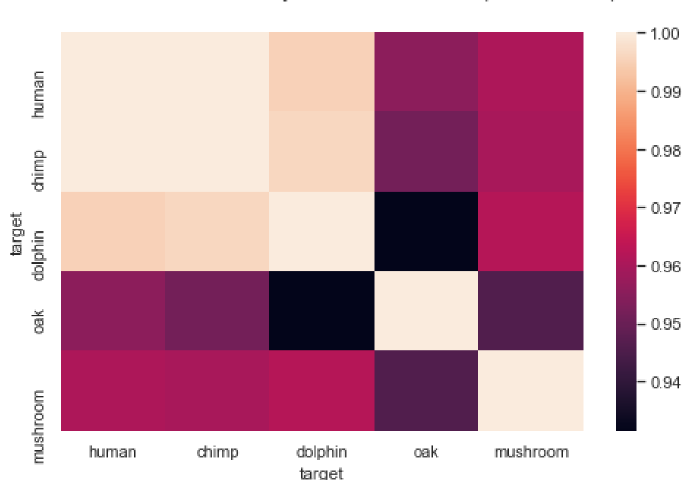
أولاً، دعنا نلقي نظرة على القيم المتوسطة لكل مجموعة عمود من خلال الحمض النووي لخمسة أنواع التي أضفتها إلى إطار البيانات هذا كمثال.

	AAA	AAC	AAG	AAT	ACA	ACC	ACG	ACT	AGA	AGC	...	TCG	TCT	TGA	TGC	TGG	TGT	TTA	TTC	TTG	TTT
	mean	mean	mean	mean	mean	mean	mean	mean	mean	mean	...	mean	mean	mean	mean	mean	mean	mean	mean	mean	mean
target																					
human	60.229400	33.362550	58.012250	34.190950	42.902200	35.12250	12.653950	31.898350	61.794450	46.252500	...	11.310700	35.77230	47.596600	40.074900	51.249600	33.977450	21.301150	35.017850	33.329400	33.42700
chimp	67.194300	37.500850	65.028200	37.705900	48.163300	40.80795	14.604500	35.642400	69.796600	52.699600	...	12.945650	39.50585	53.729300	45.264000	58.448150	38.660100	23.541550	38.992100	37.130300	36.00340
dolphin	47.661000	29.020600	49.433050	26.596250	36.743850	33.69755	15.967300	27.132200	53.910250	44.904550	...	13.051750	31.85410	40.071600	37.548450	47.272250	29.036600	17.242750	31.348700	27.255200	28.29850
oak	46.520700	23.288300	41.873250	35.526600	26.295850	16.25395	7.664500	22.174450	39.199150	21.407250	...	9.253550	28.96365	37.352750	24.595600	35.381750	26.230500	22.769400	30.478900	40.989250	37.80160
mushroom	31.000975	24.464066	29.259715	23.088972	25.872618	23.17802	21.090998	19.591673	27.279295	22.428207	...	24.247187	26.01928	22.885971	24.272693	28.102626	19.113878	13.412528	28.245236	26.103601	23.72078

5 rows × 64 columns



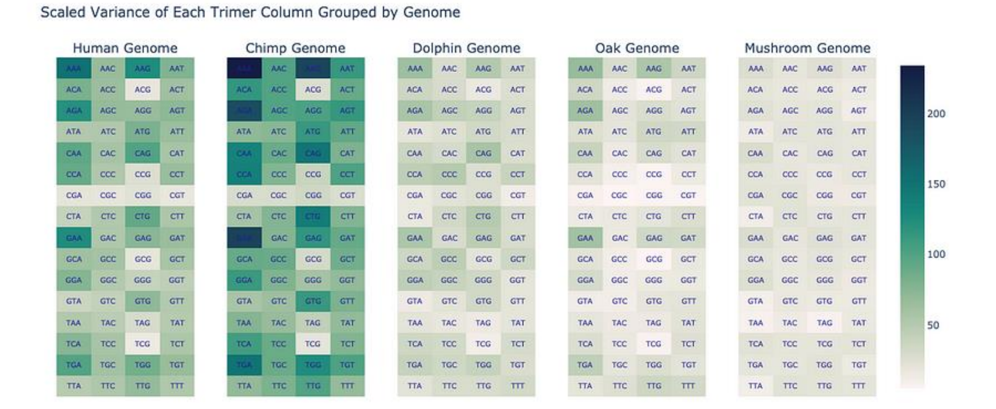
في هذا الرسم البياني، نتخيل تلك القيم المتوسطة لكل عمود مرتبة على شكل شبكة ثنائية الأبعاد. كل مخطط فرعي هو أحد الأنواع الأربعة، والآن يمكننا بصريًا مقارنة وسائل قيمهم معًا. يبدو أن بعض الأنماط تظهر بالفعل. بالنسبة لجينومات الثدييات mammalian genomes (الإنسان والشمبانزي والدلفين)، هناك الكثير من المناطق المتشابهة بصريًا. حتى أن بعض أجزاء جينوم البلوط مرتبطة بالأجزاء الثلاثة الأخرى، ولكن بالنسبة للجزء الأكبر، من السهل التمييز بالفعل أنه يمكن أن يكون مختلفًا بشكل كبير.



لقد رأينا بصريًا التشابه بين ميزاتنافي الفئات المختلفة، ولكن يمكنني أيضًا حساب تشابه جيب التمام cosine similarity بين القيم المتوسطة لكل فئة. لاحظ أن الجزء السفلي من هذا المقياس لا يزال

يمثل قيمة عالية للتشابه (> 90٪). يبدو أن جميع فئاتنا لديها تشابه كبير في جيب التمام، ولكن بالطبع، يرى الحمض النووي للإنسان والشمبانزي أعلى مستوى من التشابه.

دعنا الآن نلقي نظرة على تصور التباين variance في ميزات كل جينوم.

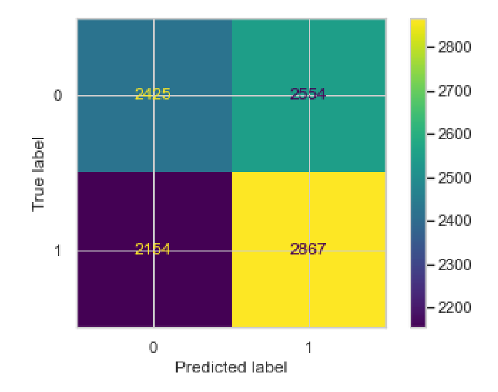


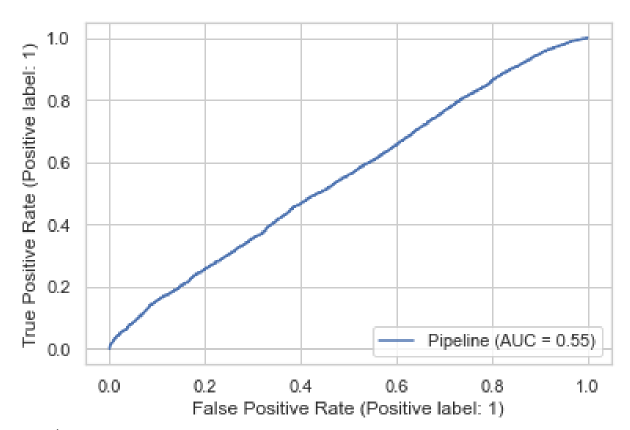
بناءً على هذا التصور، يمكننا أن نرى أن بعض الجينومات لها تباين أعلى من غيرها. سيكون من المثير للاهتمام معرفة ما إذا كان لهذا تأثير على النماذج متعددة الطبقات أو يمكن تفسيره أثناء التصنيف الثنائي لفئتين لهما مستويات مختلفة من التباين.

المصنف الثنائي للحمض النووي البشري مقابل الشمبانزي

يمكننا أن نفترض أن الحمض النووي للإنسان والشمبانزي يشبه إلى حد كبير الحمض النووي للإنسان والبلوط من اختبارات التشابه الأولية لتحليل البيانات الاستكشافية EDA وجيب التمام. سأحاول تشغيل مصنف الانحدار اللوجستي logistic regression classifier على الحمض النووي للإنسان والشمبانزي فقط.



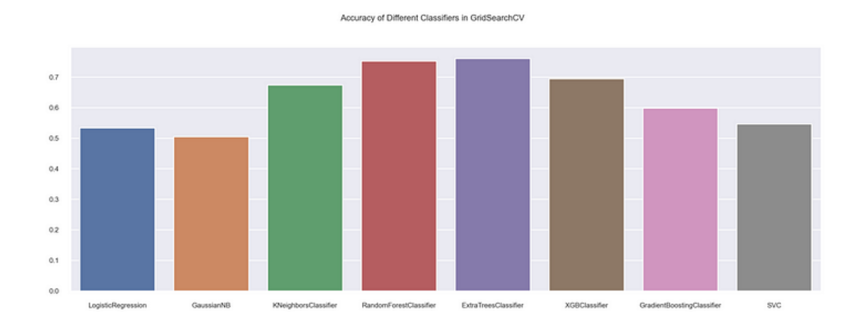




الانحدار اللوجيستي غير المعدل untuned Logistic Regression ليس دقيقًا على الإطلاق. درجة الدقة بنسبة 53٪ هي بالكاد أكثر من مجرد فرصة عشوائية لأن الفئتين في مجموعة الاختبار متوازنة. قد يكون هذا بسبب العلاقة الخطية العالية لميزاتنا. نظرًا لأنني ملتزم بمجموعة الميزات هذه، على الأقل في الوقت الحالي، سأجرب مصنفات مختلفة قد تعمل بشكل أفضل مع علاقة خطية عالية.

البحث عن أفضل مصنف ثنائي

نظرًا لأن الانحدار اللوجستي لا يبدو أنه مصنف جيد لبياناتنا، فسوف أقوم بتنفيذ بحث شبكي grid search للعديد من المصنفات المختلفة لمعرفة ما إذا كان بإمكاننا العثور على واحد يناسب البيانات بشكل أفضل. لا يزال النموذج يستخدم نفس أدوات اقتطاع بيانات data trimers التدريب والاختبار (k = 3) من KmerTransformer. سأفترض أنه مع k أعلى، سنرى درجة دقة أفضل، لكن الأمر سيستغرق أيضًا وقتًا أطول لتناسب كل نموذج. إذا وجد بحث الشبكة هذا مصنفات جيدة، فيمكنني زيادة k في بحث شبكة آخر، ولكن على مجموعة أصغر من المعلمات. أولاً، دعونا نجرب المصنفات غير المعدلة.



البحث الشبكي على المصنفات المختلفة

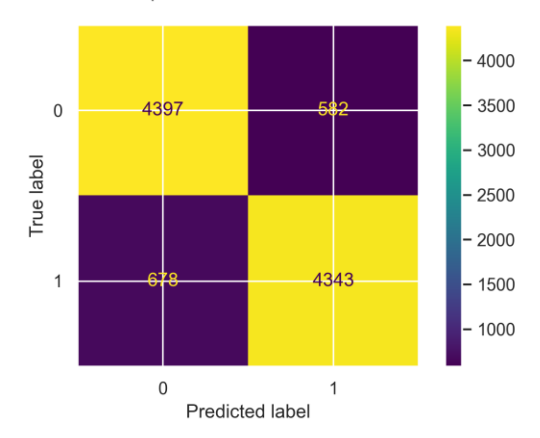
```
human_chimp_model = Pipeline(steps=[
    ('kmer_transformer', KmerTransformer(k=6,
verbose=False)),
    ('scaler', StandardScaler()),
    ('classifier', ExtraTreesClassifier())
])

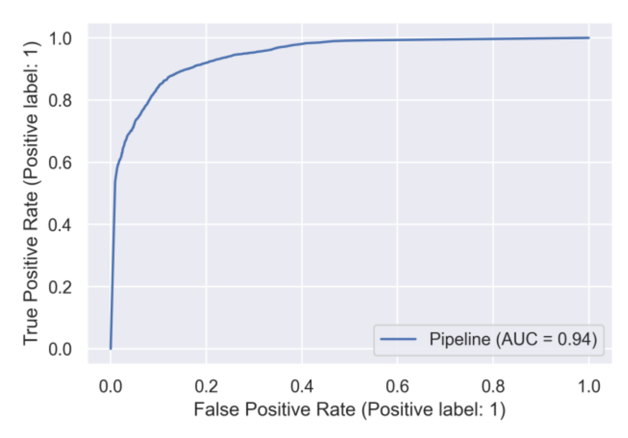
human_chimp_model.fit(X_train, y_train)
y_preds = human_chimp_model.predict(X_test)

print(classification_report(labeler.inverse_transform(y_test),
labeler.inverse_transform(y_preds))precision    recall  f1-score
support

       chimp       0.87      0.88      0.88      4979
       human       0.88      0.87      0.88      5021

    accuracy                           0.88     10000
   macro avg       0.88      0.88      0.88     10000
weighted avg       0.88      0.88      0.88     10000
```





هذه ليست نتيجة سيئة مقارنة بنموذجنا الأساسي البالغ 53٪. الدقةفي هذا النموذج هي 88٪ لتصنيف الحمض النووي للإنسان والشمبانزي. تبلغ درجة 94٪ AUC وتحتضن الزاوية اليسرى العلوية من الرسم البياني. هل يمكننا تحسين هذه النتيجة بضبط المعلمة الفائقة hyperparameter tuning؟ بالتأكيد. ولكن دعنا نحاول إدخال خوارزمية جديدة للمعالجة المسبقةفي القسم التالي لمعرفة ما إذا كان بإمكاننا تحسين خوارزمية تحويل الميزات أولاً.

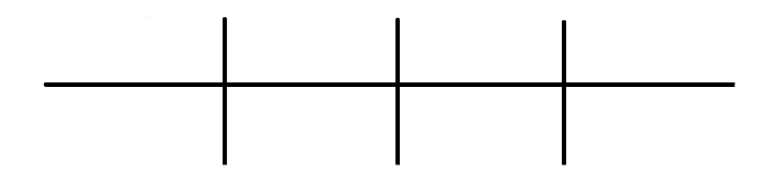
محول K-Group K-mer

هنا سوف أقوم بإنشاء محول transformer مخصص آخر للمعالجة المسبقة للبيانات. على غرار محول KmerTransformer، سيقوم هذا المحول بإنشاء قواميس تردد لكل صف من البيانات. على عكس KmerTransformer، فإنه سيجمع جميع أعداد A و C و G و T بدلاً من تسلسلها الفريد.

بعبارة أخرى، إذا كان لدينا تسلسل مثل "AAGTCGAGT" سيكون لدينا 3 مجموعات A و 1 C و 3 G و 2 T. لذلك سوف يملأ التسلسل عمودًا باسم: "A2C1G3T2". وبالمثل، فإن تسلسلًا مثل "GTGAAACTG" يملأ نفس العمود أيضًا. هذا يعني أننا سننشئ عددًا أقل من الأعمدة لنفس k. وبالتالي، سنزيد k بشكل كبير عندما نستدعي المحول.

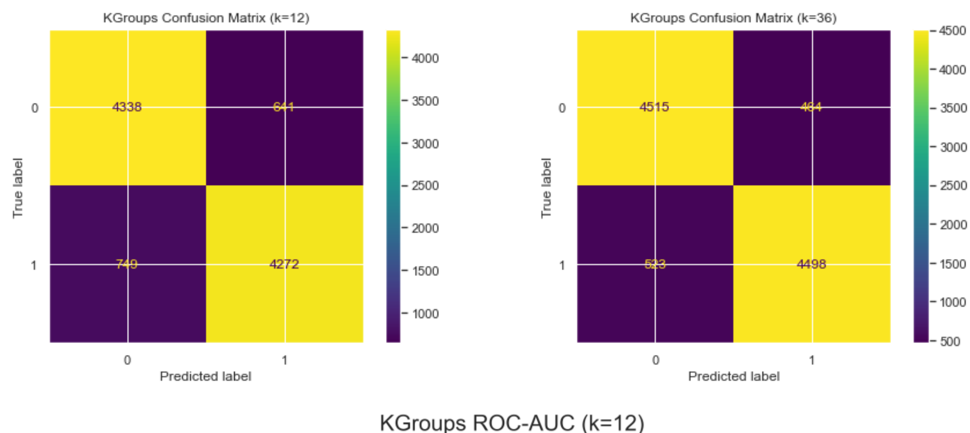
KGroupKmerTransformer(k=3)

ACGTCGTACG

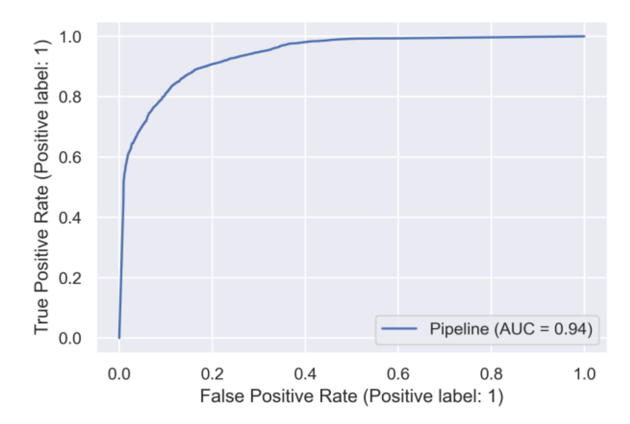


يمكن أيضًا العثور على فئة المحولات هذه على صفحة GitHub الخاصة بي.

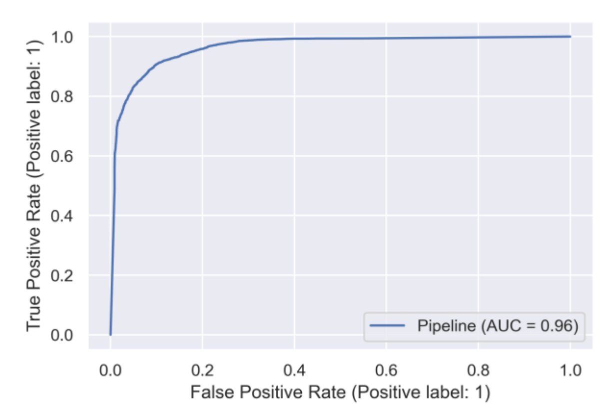
باستخدام فئة KgroupKMerTransformer المخصصة، دعنا نجربها على البيانات باستخدام k = 12 و k = 36.



KGroups ROC-AUC (k=12)







نحصل الآن على دقة 90٪ عند مقارنة الحمض النووي للإنسان والشمبانزي و 36 = k. بعد تقليم pruning الأعمدة التي لم يتم ملؤها مطلقًا، استخدمنا 7386 عمودًا إجماليًا. تبلغ درجة AUC لدينا الآن 96٪ وتبدو أفضل مما كانت عليهفي نموذج محول k-mer السابق.

حتى الآن، جيد جداً — لنجرب هذا النموذج على بيانات الفئات المتعددة Multi-Class data.

تصنيف متعدد الفئات

الآن سنطبق KgroupKMerTransformer على بيانات متعددة الفئات مع k = 36.

```
Multiclass Performance with Kgroup Transformer (k=36)
precision recall f1-score support

chimp 0.91 0.83 0.87 5050
dolphin 0.90 0.87 0.89 5034
human 0.91 0.87 0.89 4964
mushroom 0.73 0.84 0.79 3225
oak 0.86 0.91 0.89 5060

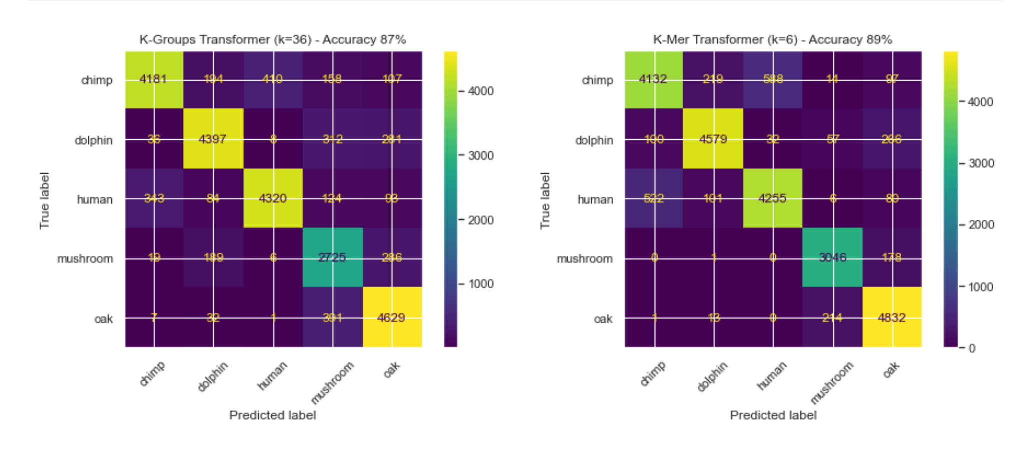
accuracy 0.87 23333
macro avg 0.86 0.87 0.86 23333
weighted avg 0.87 0.87 0.87 23333
```

يبدو أن نموذج Kgroup Multiclass أقل دقة من النموذج الثنائي. خاصةفي فئات معينة. على سبيل المثال، درجة f1 للفطر منخفضة بشكل خاص مقارنة بالآخرين. دعنا نشغل نموذج KmerTransformer الأصلي على بيانات متعدده الفئات ونقارن النتائج.

```
Multiclass Performance with Kmer Transformer (k=6)
precision recall f1-score support

chimp 0.87 0.82 0.84 5050
dolphin 0.93 0.91 0.92 5034
human 0.87 0.86 0.86 4964
mushroom 0.91 0.94 0.93 3225
oak 0.89 0.95 0.92 5060

accuracy 0.89 23333
macro avg 0.89 0.90 0.90 23333
weighted avg 0.89 0.89 0.89 23333
```



يمكننا أن نرىفي هذه المقارنة لمصفوفات الارتباك أن نموذج KmerTransformer المُعالج مسبقًا أفضل بكثيرفي تصنيف بعض أجزاء البيانات التي يواجهها نموذج Kgroup وقتًا أصعب. بالنسبة للنموذج النهائي، سنجمع بين محولي المعالجة المسبقة السابقين اللذين أنشأناهمافي نموذج واحد.

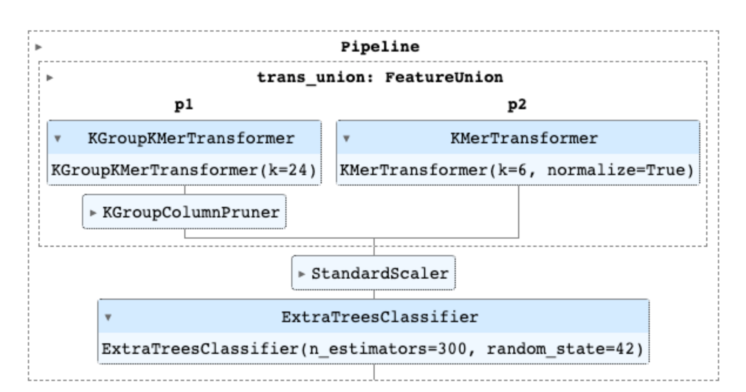
النموذج النهائي - Kgroup + Kmer

بعد البحث الشبكي لتحديد أفضل k لكل من KgroupKMerTransformer و KmerTransformer، بالإضافة إلى البحث عن المعلمات الفائقة عن أفضل ExtraTreesClassifier، سننشئ النهائي ونرى النتائج.

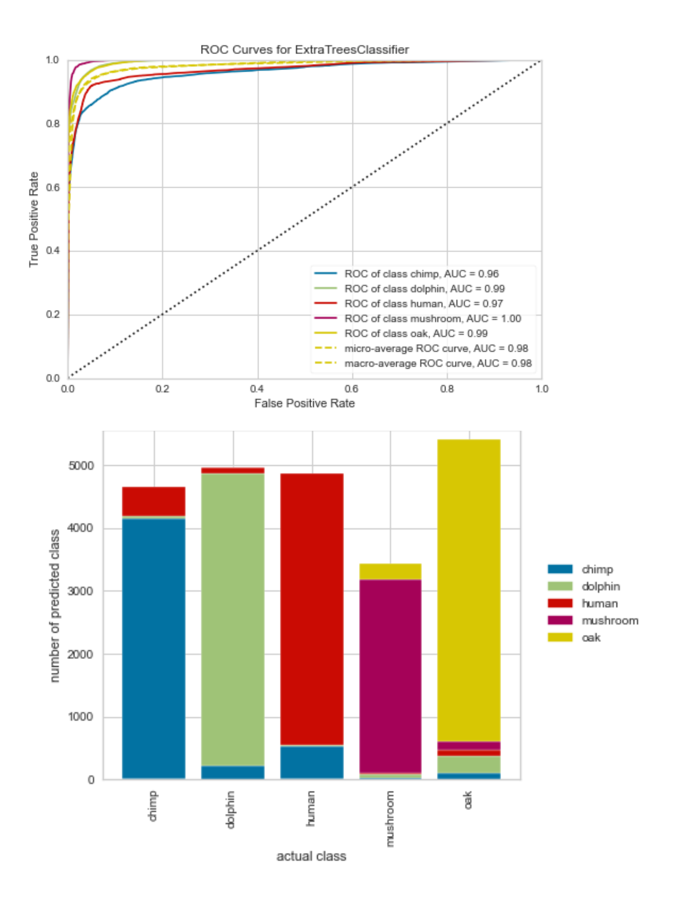
النموذج النهائي للمعلمات الفائقة:

- KgroupTransformer: k=24
- KmerTransformer: k=6
- ExtraTreesClassifier: n_estimators=300

باستخدام FeatureUnionفي sklearn داخل خط أنابيب، يمكننا أن نجمع الأعمدة المحولة من KgroupTransformer و KmerTransformer قبل تشغيلها من خلال ExtraTreesClassifier.



	precision	recall	f1-score	support
chimp	0.89	0.82	0.86	5050
dolphin	0.93	0.92	0.93	5034
human	0.89	0.87	0.88	4964
mushroom	0.90	0.96	0.92	3225
oak	0.89	0.95	0.92	5060
accuracy			0.90	23333
macro avg	0.90	0.90	0.90	23333
weighted avg	0.90	0.90	0.90	23333



النتائج

حقق نموذجي النهائي دقة 90٪ على بيانات الاختبار لتصنيف التسلسلات بين 5 جينومات من الحمض النووي. على الرغم من وجود مجال للتحسين في النموذج، بالإضافة إلى تجربة طرق التصنيف الأخرى (مثل الشبكات العصبية والتعلم العميق)، يمكنني رفض فرضيتي العدمية، ويمكننا افتراض وجود العديد من أنماط التشابه بين فئات الجينوم بيانات. علاوة على ذلك، يمكننا تمييز الأنماط والتباين وتصنيف الأنواع من تسلسلات الحمض النووي الفردية بدقة عالية.

إذا كنت مهتمًا بمعرفة المزيد عن الكود الخاص بي، فيمكنك التحقق من نوتبوك جوبيتر الكامل على صفحة GitHub الخاصة بي.

المصدر:

https://medium.com/@rollingstorms/dna-sequence-classification-for-species-prediction-df202bcb621f

17) تحديد تسلسل DNA الحقيقي مع التعلم العميق Identification of Bona Fide DNA Sequences with Deep Learning

مشروع Github: https://github.com/Jdduryea/SeqID

في الفصل الدراسي الماضي، درست دورة تعلم الآلة الإحصائية statistical machine learning في جامعة رايس Rice University. ركزنا بشكل أساسي على مجموعات بيانات الصور، بما في ذلك CIFAR-10 الشهير. كانت دورة رائعة، لكننا لم نتعلم الكثير حول كيفية التعامل مع النص text أو البيانات المتسلسلة sequential data. في نهاية هذا الأسبوع قررت اختبار المياه. أنا منفتح دائمًا على الاقتراحات، لذا إذا كان لديك أي تعليقات أو أسئلة أو مخاوف، فيرجى إبلاغي بذلك! لقد بدأت للتو في التلاعب بهذا الموضوع وأحاول أن أتعلم بقدر ما أستطيع.

بصفتي متدربًا في المعلوماتية الحيوية bioinformatics بدوام جزئي في كلية بايلور Baylor للطب (على الجانب الآخر من الشارع من رايس يو)، أعمل مع البيانات الجينية والجينية الوراثية طوال الوقت. الجينوم البشري human genome عبارة عن سلسلة من 3 مليارات زوج أساسي (A، C، G، T) ويوفر مجموعة كبيرة من البيانات للبدء. هناك العديد من الأسئلة التي يمكننا طرحها حول الجينوم، لكنني أردت أن أبدأ بسؤال بسيط: هل يمكن لنموذج التعلم العميق أن يميز الفرق بين التسلسلات الحقيقية real sequences للحمض النووي البشري والتسلسلات المزيفة المتولدة fake, generated sequences؟

الخطوة 1: جمع البيانات

في البداية كنت بحاجة لجمع بعض بيانات الحمض النووي DNA. الجينوم البشري متاح للجمهور ويسهل الوصول إليه عبر الإنترنت. يمكن الحصول على تسلسل الحمض النووي الخام من جامعة كاليفورنيا سانتا كروز California Santa Cruz (https://genome.ucsc.edu/) بالأمر التالي:

```
curl http://togows.org/api/ucsc/hg38/chr19:400,000-40,000,500.fasta > bigseq.txt
```

سيؤدي هذا إلى تنزيل بيانات تسلسل الحمض النووي من الجينوم البشري (الإصدار hg38، والذي يستخدم على نطاق واسع في التعليق التوضيحي annotation) على الكروموسوم 19 من مؤشر الزوج الأساسي 400000 إلى 40000500.

بشكل أساسي، سيؤدي هذا إلى تخزين تسلسل هائل من A و C و G و T في الملف المسمى bigseq.txt. سيبدو مثل هذا (لكن آلاف الأسطر أطول):

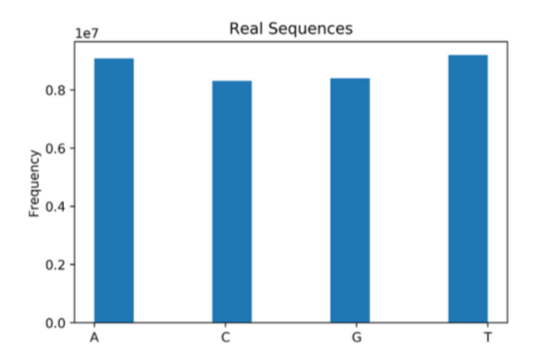
الخطوة 2: المعالجة المسبقة للبيانات

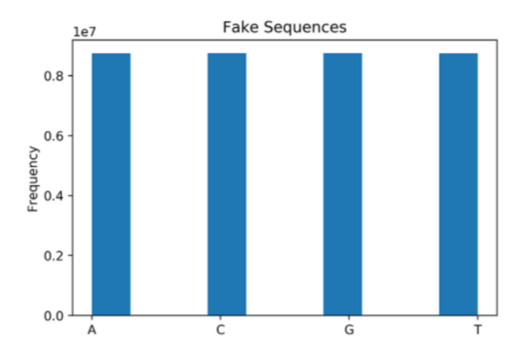
```
TTGTGGACCGAGATCCAGGTCTCAGTGTAGACCAAGGAACAGAGGCAGCCCCTACTCGGG
CTCCCAGGCTCCAGCCAGGCCTGCTAGGTGGGCGGCCTCTGCCCGTGTGATGGAAGGAAG
CCAGGCTCAGAGATGGGGTGACTTGTCCCACGTCCTGGAGGGTCTCACAGCTCCACCACG
GCCAGGAGACTTCAGTCCAGGGGGCCTCCATCCCCAGCGACCTTCACCTGACCCCCAGGG
GACCCCAGCCAACAAGACCCGGCCTGCAGCTCCGGAACGGGGGAGGGCTGCTCTCCACCG
CCCCTGTGCGGCCGCCCGGGAAAGTGCAGGCGGGCCGGGCGCGGTGGCTCACGCCTGTGA
TCTCAGCACTTTGGGAGGCCGAGGTGGGCGGATCACCTGAGGTCGGGAGTTCGAGGCCAG
CCTGCCCAACATGGAGAAACCCTGTCTCTACTAAAGATACAAAATTAGCCAGGCGTGGTG
ACGCATGCCTGTAATCCCAGCTACTGGAGTGGCTGAGGCAGGAGAATCGCTTGAGCCCGG
GAGACAGAGGTTGCGGTGAGCTGAGATCGCACCATTGCACTCCAGCCTGGGCAACAAGAG
CGAAACTCAGAAAAAAAAGAAAAGAAAGTGCAGGGGACCCGCCGTCGGGGTGGGGGCGGC
GCTGCCCAGCCTCTGTCCCACTTCCATGCACTTGACCTCGACCCTCCGGCCTCCGTCTGC
GATCTTCCCGTGCCTGAATATGAGGCTTGGAACAGACCCAGACCTTCCTGCCTGCCCGTC
CTGAGTGGCCCCGGGACCCCGCCCCATCTTTGGCCCCCAGCCCCTGCCTCTCTGCCGCCT
```

كان هذا هو الجزء الجديد بالنسبة لي. قد يكون العمل مع البيانات النصية أمرًا صعبًا، لكن لحسن الحظ لدينا فقط 4 رموز (1 لكل زوج أساسي base pair) للعمل معها. نظرًا لأن نماذج التعلم العميق لا يمكنها العمل مع هذا النوع من البيانات بشكل مباشر، فقد احتجت أولاً إلى تحويلها إلى بيانات رقمية. لقد قمت بتمثيل A بـ 1، C مع 2، G مع 3، و T مع 4. لقد استخلصت 3500 تسلسل فرعي، طول كل منها 10000 زوج أساسي، وقمت بتحويل البيانات إلى تنسيق رقمي، وأعطيت كل نقطة من نقاط البيانات هذه التسمية 1 لقد قمت بشكل عشوائي بإنشاء 3500 تسلسل بطول 10000 وأعطيت هذه التسمية 0. المزيد عن هذافي (https://github.com/Jdduryea/SeqID) Github repo. قمت بتقسيم البيانات إلى بيانات التدريب والتحقق من الصحة (تقسيم 20/80) باستخدام SKLearn.

الخطوة 3: استكشاف البيانات والتصور

استكشاف البيانات الخاصة بك أمر ضروري دائمًا. ألقيت نظرة على توزيع الأزواج الأساسيةفي البيانات الحقيقية والمزيفة.





يبدو أن كلا مجموعتي البيانات لهما توزيع منتظم uniform distribution نسبيًا للأزواج الأساسية base pairs. سيواجه النموذج وقتًا صعبًافي التمييز بين البيانات الحقيقية والمزيفة بمجرد النظر إلى الأعداد النسبية لكل زوج أساسيفي تسلسل معين.

الخطوة 4: إنشاء النموذج

حاولت إنشاء نموذج ذاكرة طويلة قصيرة المدى long short-term memory (LSTM)، لكن هذا استغرق وقتًا طويلاً للتدريب على الكمبيوتر المحمول الخاص بي. لقد جربت أيضًا الشبكة امامية التغذية الاساسية vanilla feed forward network، لكن الأداء كان سيئًا. كان الانحدار اللوجستي logistic regression البسيطفي البيانات أسوأ. كدت أتخلى عن المشروع، معتقدة أنه ربما لم تكن المشكلة قابلة للحل، لكنني قررت أن أعطيها فرصة أخيرة.

اخترت نموذج تلافيفي احادي البعد 1-d convolutional model مبني باستخدام Keras. الفكرة الأساسية للالتفاف احادي البعد 1-d هي أن الطبقة الأولى من النموذج تنظر إلى جزء صغير من الحمض النووي وترسل إشارة اعتمادًا على ما رآه. ثم ينتقل إلى الجزء الصغير التالي ويفعل الشيء نفسه. تستمر هذه العملية حتى "انزلقت slid " الطبقة الأولى أو "ملتفة convolved" فوق تسلسل الإدخال. تلتقط الطبقة الثانية الإشارة من الطبقة الأولى وتنفذ روتينًا مشابهًا، ولكن هذه المرة يكون تمثيل البيانات أكثر تجريدًا نظرًا لأنه يلتف عبر هذه الإشارات الجديدة. بعد ذلك، تحدد طبقة التجميع pooling layer المناطق ذات أقوى إشارات الإخراج وترسل تلك الإشارات إلى الأمام.

```
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_129 (Conv1D)          (None, 9999, 16)          48

conv1d_130 (Conv1D)          (None, 9998, 16)          528

max_pooling1d_31 (MaxPooling (None, 3332, 16)          0

conv1d_131 (Conv1D)          (None, 3330, 16)          784

global_average_pooling1d_26  (None, 16)                0

dense_164 (Dense)            (None, 100)               1700

dropout_94 (Dropout)         (None, 100)               0

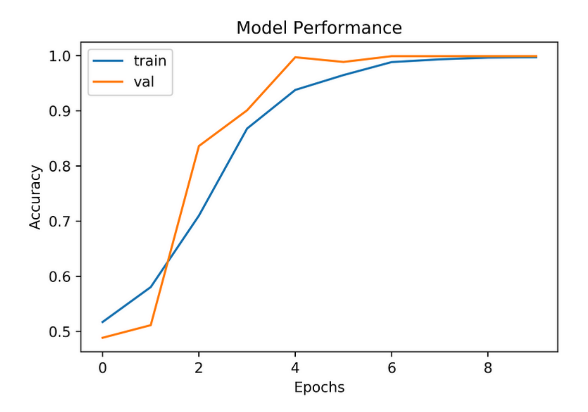
dense_165 (Dense)            (None, 1)                 101
=================================================================
Total params: 3,161
Trainable params: 3,161
Non-trainable params: 0
_________________________________________________________________
None
```

لقد قمت بتكديس بضع طبقات تلافيفية convolutional layers بأحجام فلتر صغيرة، وبعض التجميعات pooling، وطبقة كثيفة مخفية hidden dense layer لبعض التثليج على الكعكة. لقد استخدمت إنتروبيا متقاطعة ثنائية لدالة الخطأ binary cross entropy، ومحسن ADAM، وتنشيطات ReLU.

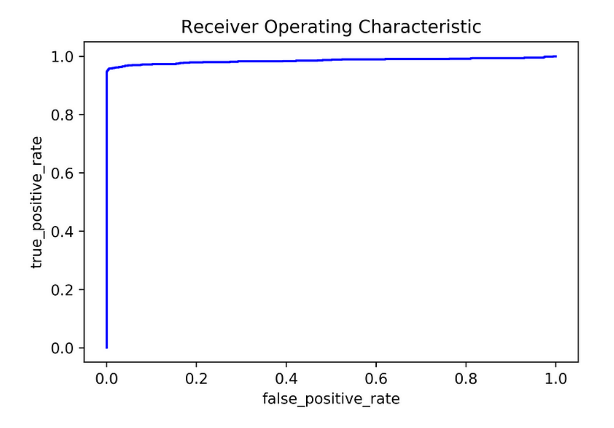
أداء النموذج

بعد قليل من التدريب على i7 المتواضع لجهاز الكمبيوتر المحمول، كنت سعيداً جداً بالنتائج. كان أداء النموذج جيداًفي بيانات التدريب والتحقق من الصحة! ما يقرب من 100٪ دقة! لقد حملت مجموعة بيانات الاختبار النهائية خلال مرحلة التدريب / التحقق من الصحة واختبرت النموذج على مجموعة البيانات النهائية هذه، حيث حققت دقة 99.714٪.

يمكننا أن نرى مدى جودة أداء النموذج أثناء تدريبه بمرور الوقت. يكون منحنى بيانات التدريب سلسًا جداً، بينما يأخذ منحنى بيانات التحقق من الصحة بعض التقلبات الحادة، لكن كلاهما يتقارب مع دقة مثالية تقريبًا. لم تكن هناك فترات epochs قليلة أخرى لتؤذي نظرًا لأن دقة التحقق كانت لا تزال تتزايد عند بلوغ الحد الأقصى للفترة.



يعرض النموذج مقايضة جيدة بين الإيجابيات الحقيقية true positives والإيجابيات الزائفة false positives. أي أنه لا يبدو أنه يفضل فئة على الأخرى.



الاستنتاجات

ركز هذا المنشور على مشكلة بسيطة: معرفة الفرق بين الحمض النووي الحقيقي والمحاكى (المزيف). يمكن تمييز هذا الحمض النووي الحقيقي عن الحمض النووي المزيف يدعم الحقيقة المعروفة بأن الحمض النووي لدينا منظم للغاية وليس مجرد ضوضاء عشوائية. للمضي قدمًافي التعلم الشخصي الخاص بي، بالإضافة إلى بحثيفي المعلوماتية الحيوية، أود معالجة المزيد من المشكلات المثيرة للاهتمام والاستفادة من الطبيعة عالية التنظيم للحمض النووي.

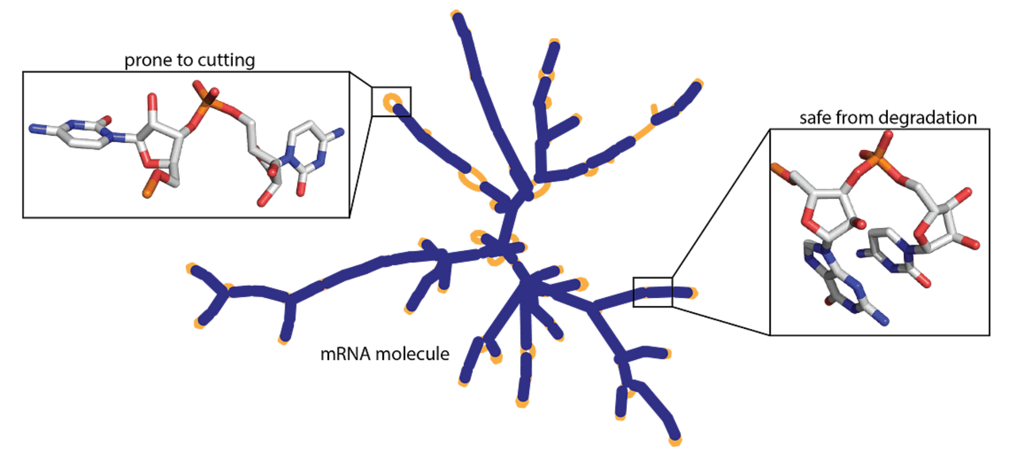
المصدر

https://medium.com/@jackduryea/identification-of-bona-fide-dna-sequences-with-deep-learning-9dee0081a5a8

18) نموذج التعلم العميق للتنبؤ بتحلل mRNA Deep learning model to predict mRNA degradation

تصميم نموذج التعلم العميق الذي سيتنبأ بمعدلات التحلل degradation rates في كل قاعدة لجزيء الحمض النووي الريبي RNA باستخدام مجموعة بيانات Eterna التي تضم أكثر من 3000 جزيء من RNA.

لقاحات mRNA في طليعة مكافحة جائحة COVID-19 وهي مصحوبة بقيود. تحد مشكلة الاستقرار في جزيئات RNA المرسال messenger RNA (mRNA) من حزمها في محقنة يمكن التخلص منها وتوزيعها حول العالم باستخدام نظام التبريد refrigerating system (nih.gov). يتمثل التحدي الرئيسي في تصميم لقاح مستقر لـ mRNA يمكنه تحمل الشحن في جميع أنحاء العالم لأن قطع واحد يمكن أن يجعل اللقاح بأكمله عديم الفائدة. اكتشف الباحثون أيضًا أن جزيئات mRNA تميل إلى التحلل degrade بسرعة، وفي هذا المشروع، سنقوم بتصميم النموذج للتنبؤ بمعدل التحلل degradation rate الذي يمكن أن يساعد العلماء والباحثين على تصميم لقاحات أكثر استقرارًا في المستقبل. حاليًا، للتغلب على هذه المشكلة، نحتفظ بهذه اللقاحات تحت التبريد المكثف، لكن هذا أيضًا محدود لأن هذه اللقاحات متاحة لعدد أقل من الناس حول العالم. OpenVaccine



أهداف المشروع

في هذا المشروع، سنقوم باستكشاف مجموعة البيانات الخاصة بنا ثم التسلسل المُعالج مسبقًا، والهيكل، وميزات نوع الحلقة المتوقعة بحيث يمكن استخدامها لتدريب نموذج GRU للتعلم العميق. أخيرًا توقع سجلات التحلل degradation records على مجموعات البيانات العامة واختبارها.

الاستعداد

سنستخدم TensorFlow كمكتبتنا الرئيسية لبناء وتدريب نموذجنا و Pandas / JSON لاستيعاب البيانات. للتصور، سنستخدم Plotly وللتلاعب بالبيانات Numpy.

```
# Dataframe
import json
import pandas as pd
import numpy as np# Visualization
import plotly.express as px# Deeplearning
import tensorflow.keras.layers as L
import tensorflow as tf# Sklearn
from sklearn.model_selection import train_test_split#Setting seeds
tf.random.set_seed(2021)
np.random.seed(2021)
```

معلمات التدريب

- **Target columns**: التفاعلية، deg_Mg_pH10، deg_Mg_50C، deg_pH10، deg_50C
- **Model_Train**: صحيح إذا كنت تريد تدريب نموذج يستغرق ساعة واحدة للتدريب.

```
# This will tell us the columns we are predicting
target_cols = ['reactivity', 'deg_Mg_pH10', 'deg_Mg_50C', 'deg_pH10',
'deg_50C']
Model_Train = True # True if you want to Train model which take 1 hour
to train.
```

مقياس أداء النموذج الخاص بنا هو MCRMSE (متوسط جذر العمود متوسط الخطأ التربيعي Mean column-wise root mean squared error)، والذي يأخذ الجذر التربيعي للخطأ التربيعي للحقيقة الأساسية لجميع الأعمدة المستهدفة.

$$\text{MCRMSE} = \frac{1}{N_t}\sum_{j=1}^{N_t}\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_{ij}-\hat{y}_{ij})^2}$$

حيث عدد أعمدة هدف الحقيقة الاساسية المسجلة، وهي القيم الفعلية actual والمتوقعة predicted values، على التوالي.

```
def MCRMSE(y_true, y_pred):## Monte Carlo root mean squared errors
colwise_mse = tf.reduce_mean(tf.square(y_true - y_pred), axis=1)
return tf.reduce_mean(tf.sqrt(colwise_mse), axis=1)
```

تتوفر بيانات تحلل mRNA على Kaggle.

شرح تفاصيل الأعمدة

- **Id**: عينة معرف فريد.
- **seq_scored**: يجب أن يتطابق هذا مع طول أعمدة التفاعلية و *_deg وأعمدة error.
- **seq_length**: طول التسلسل.

- **sequence**: يصف تسلسل RNA، وهو مزيج من A و G و U و C لكل عينة.
- **structure**: مصفوفة من (،) و. الأحرف التي يتم التبرع بها للقاعدة هي أن يتم إقرانها paired أو إلغاء إقرانها unpaired.
- **reactivity**: هذه الأرقام هي قيم تفاعلية لأول 68 قاعدة تستخدم لتحديد البُنية الثانوية المحتملة لعينة RNA.
- **deg_pH10**: احتمالية التحلل بعد الحضانة بدون المغنيسيوم على الرقم الهيدروجيني 10 عند القاعدة base أو الوصلة linkage.
- **deg_Mg_pH10**: احتمالية التحلل بعد الحضانة بالمغنيسيوم على الرقم الهيدروجيني 10 عند القاعدة أو الوصلة.
- **deg_50C**: احتمال التحلل بعد الحضانة بدون المغنيسيوم عند 50 درجة مئوية عند القاعدة أو الوصلة.
- **deg_Mg_50C**: احتمالية التحلل بعد الحضانة بالمغنيسيوم عند 50 درجة مئوية عند القاعدة أو الوصلة.
- ***_error_***: الأخطاء المحسوبةفي القيم التجريبية التي تم الحصول عليهافي التفاعلية، وأعمدة *_deg.
- **Forecasted_loop_type**: أنواع الحلقة التي يحددها bpRNA من Vienna والتي تقترح، S: paired Stem, M: Multiloop, I: Internal loop, B: Bulge, H: Hairpin loop, E: dangling End, X: external loop.

مراقبة البيانات

```
data_dir = "stanford-covid-vaccine/"
train = pd.read_json(data_dir + "train.json", lines=True)
test = pd.read_json(data_dir + "test.json", lines=True)
sample_df = pd.read_csv(data_dir + "sample_submission.csv")
```

لدينا تسلسل وبُنية وأنواع حلقات متوقعةفي تنسيقات نصية. سنقوم بتحويلها إلى رموز رقمية بحيث يمكن استخدامها لتدريب نماذج التعلم العميق. ثم لدينا مصفوفات داخل الأعمدة من reactivity_error إلى deg_50C سنستخدمها كأهداف.

```
train.head(2)print('Train shapes: ', train.shape) print('Test shapes: 
', test.shape)Train shapes:  (2400, 19)
Test shapes:  (3634, 7)
```

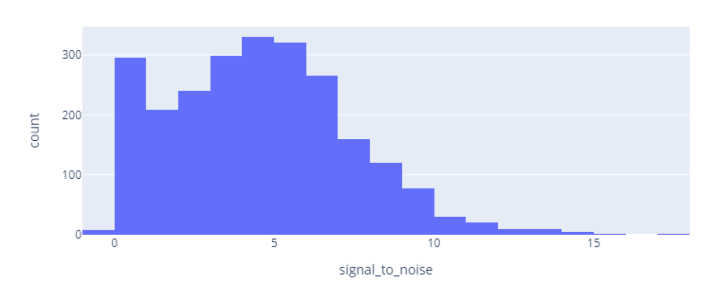
تحتوي مجموعة بيانات الاختبار فقط على التسلسل sequence والهيكل structure ونوع الحلقة المتوقع predicted_loop_type وطول السلسلة seq_length والمتسلسلة seq_scored والمعرف id. سنستخدم مجموعة بيانات اختبار للتنبؤ بمعدل التحلل لدرجة لوحة القائد العام public leader board score.

توزيع الإشارة إلى الضوضاء

يمكننا أن نرى أن توزيع الإشارة إلى الضوضاء signal-to-noise distribution يتراوح بين 0 إلى 15 وأن غالبية العينات تقع بين 0ـ6. لدينا أيضًا قيم سالبة علينا التخلص منها.

```
fig = px.histogram(
train,
"signal_to_noise",
nbins=25,
title='signal_to_noise distribution',
width=800,
height=400
)
fig.show()
```

signal_to_noise distribution




```
train = train.query("signal_to_noise >= 1")
```

طول تسلسل الاختبار

بعد النظر إلى توزيع طول التسلسل، نعلم أن لدينا طولين تسلسلين مميزين، أحدهما عند 107 والآخر عند 130.

```
fig = px.histogram(
test,
"seq_length",
nbins=25,
title='sequence_length distribution',
width=800,
height=400
)
fig.show()
```

sequence_length distribution

count

seq_length

تجزئة الاختبار إلى إطار بيانات عام وخاص

دعنا نقسم مجموعة بيانات الاختبار الخاصة بنا على أساس طول التسلسل. سيؤدي القيام بذلك إلى تحسين الأداء العام لنموذج GRU الخاص بنا.

```
public_df = test.query("seq_length == 107")private_df =
test.query("seq_length == 130")
```

إنشاء حرف إلى قاموس أعداد صحيحة سنستخدمه لتحويل تسلسل RNA والبنية ونوع الحلقة التنبؤية إلى أعداد صحيحة.

```
token2int = {x: i for i, x in enumerate("().ACGUBEHIMSX")}
token2int{'(': 0, ')': 1, '.': 2, 'A': 3, 'C': 4, 'G': 5, 'U': 6, 'B':
7, 'E': 8, 'H': 9, 'I': 10, 'M': 11, 'S': 12, 'X': 13}
```

تحويل إطار البيانات الى مصفوفة ثلاثية الابعاد

تأخذ الدالة أدناه إطار بيانات Pandas وتحولها إلى مصفوفة NumPy ثلاثية الأبعاد. سنستخدمه لتحويل ميزات التدريب والأهداف.

```
def dataframe_to_array(df):
return np.transpose(np.array(df.values.tolist()), (0, 2, 1))
```

ترميز التسلسل

Tokenization of Sequence

تستخدم الدالة أدناه سلسلة من قاموس الأعداد الصحيحة التي أنشأناها مبكرًا لتحويل ميزات التدريب إلى مصفوفات تحتوي على أعداد صحيحة. ثم سنستخدم dataframe_to_array لتحويل مجموعة البيانات الخاصة بنا إلى مصفوفة NumPy ثلاثية الأبعاد.

```
def dataframe_label_encoding(
df, token2int, cols=["sequence", "structure", "predicted_loop_type"]
):
return dataframe_to_array(
df[cols].applymap(lambda seq: [token2int[x] for x in seq])
) ## tokenization of Sequence, Structure, Predicted loop
```

المعالجة المسبقة للميزات والتسميات

- استخدام دالة الترميز على التسمية label encodingفي ميزات التدريب لدينا.
- تحويل إطار البيانات الهدف إلى مصفوفة ثلاثية الأبعاد.

```
train_inputs = dataframe_label_encoding(train, token2int) ## Label
encoding
train_labels = dataframe_to_array(train[target_cols]) ## dataframe to
3D array to
```

تقسيم التدريب والتحقق من الصحة

تقسيم بيانات التدريب لدينا إلى مجموعات تدريب والتحقق من الصحة. نحن نستخدم فلتر إشارة إلى ضوضاء noise filter لتوزيع مجموعة البيانات الخاصة بنا بالتساوي.

```
x_train, x_val, y_train, y_val = train_test_split(
train_inputs, train_labels, test_size=0.1, random_state=34,
stratify=train.SN_filter
)
```

المعالجة المسبقة لإطار البيانات العام والخاص

في وقت سابق، قمنا بتقسيم مجموعة بيانات الاختبار الخاصة بنا إلى عامة وخاصة بناءً على طول التسلسل، والآن سنستخدم dataframe_label_encoding لترميزها وإعادة تشكيلهافي مصفوفة NumPy كما فعلنا نفس الشيء مع مجموعة بيانات التدريب.

```
public_inputs = dataframe_label_encoding(public_df, token2int)
private_inputs = dataframe_label_encoding(private_df, token2int)
```

التدريب/تقييم النموذج

قبل القفز مباشرة إلى نموذج التعلم العميق، اختبرنا معززات التدرج gradient boosts الأخرى مثل Light GBM وCatBoost. أثناء تعاملنا مع التسلسل، جربت نماذج BiLSTM، لكن أداءها جميعًا كان أسوأ مقارنة بنموذج GRU الثلاثي مع التنشيط الخطي linear activation.

يتأثر هذا النموذج بنماذج xhlulu الأولية، وقد اندهشت من مدى بساطة طبقة GRU في تحقيق أفضل النتائج الممكنة دون استخدام زيادة البيانات data augmentation أو هندسة الميزات feature engineering.

لمعرفة المزيد حول RNNs وLSTM وGRU، يرجى الاطلاع على منشور المدونة هذا.

```
def build_model(
embed_size, # Length of unique tokens
seq_len=107, # public dataset seq_len
pred_len=68, # pred_len for public data
dropout=0.5, # trying best dropout (general)
sp_dropout=0.2, # Spatial Dropout
embed_dim=200, # embedding dimension
hidden_dim=256, # hidden layer units
):
inputs = L.Input(shape=(seq_len, 3))
embed = L.Embedding(input_dim=embed_size,
output_dim=embed_dim)(inputs)
reshaped = tf.reshape(
embed, shape=(-1, embed.shape[1], embed.shape[2] * embed.shape[3])
)
hidden = L.SpatialDropout1D(sp_dropout)(reshaped)
# 3X BiGRU layers
hidden = L.Bidirectional(
L.GRU(
hidden_dim,
dropout=dropout,
return_sequences=True,
kernel_initializer="orthogonal",
)
)(hidden)
```

```
hidden = L.Bidirectional(
L.GRU(
hidden_dim,
dropout=dropout,
return_sequences=True,
kernel_initializer="orthogonal",
)
)(hidden)
hidden = L.Bidirectional(
L.GRU(
hidden_dim,
dropout=dropout,
return_sequences=True,
kernel_initializer="orthogonal",
)
)(hidden)
# Since we are only making predictions on the first part of each
sequence,
# we have to truncate it
truncated = hidden[:, :pred_len]
out = L.Dense(5, activation="linear")(truncated)
model = tf.keras.Model(inputs=inputs, outputs=out)
model.compile(optimizer="Adam", loss=MCRMSE) # loss function as of
Eval Metric
return model
```

بناء النموذج

بناء نموذجنا عن طريق إضافة حجم التضمين embed size (14) وسنستخدم القيم الافتراضية للمعلمات الأخرى.

- طول التسلسل sequence length: 107
- طول التنبؤ prediction length: 68
- التسرب dropout: 0.5
- التسرب المكاني spatial dropout: 0.2
- الأبعاد المضمنة embedded dimensions: 200
- أبعاد الطبقات المخفية hidden layers dimensions: 256

```
model = build_model(
    embed_size=len(token2int) ## embed_size = 14
)  ## uniquie token in sequence, structure, predicted_loop_type
model.summary()Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 107, 3)]          0
_________________________________________________________________
embedding (Embedding)        (None, 107, 3, 200)       2800
_________________________________________________________________
tf.reshape (TFOpLambda)      (None, 107, 600)          0
_________________________________________________________________
```

```
spatial_dropout1d (SpatialDr (None, 107, 600)          0
_________________________________________________________________
bidirectional (Bidirectional (None, 107, 512)          1317888
_________________________________________________________________
bidirectional_1 (Bidirection (None, 107, 512)          1182720
_________________________________________________________________
bidirectional_2 (Bidirection (None, 107, 512)          1182720
_________________________________________________________________
tf.__operators__.getitem (Sl (None, 68, 512)           0
_________________________________________________________________
dense (Dense)                (None, 68, 5)             2565
=================================================================
Total params: 3,688,693
Trainable params: 3,688,693
Non-trainable params: 0
_________________________________________________________________
```

تدريب النموذج

سنقوم بتدريب نموذجنا لمدة 40 فترة epochs وحفظ model checkpoint في مجلد النموذج. لقد جربت أحجام الدُفعات batch sizes من 16، 32، 64 وإلى حد بعيد 64 حجم دفعة أنتجت نتائج أفضل وتقارب سريع fast convergence.

كما يمكننا أن نلاحظ أن كلاً من خطأ التدريب والتحقق من الصحة (MCRMSE) يتناقص مع كل تكرار حتى 20 فترة ومن هناك يبدأون في التباعد diverge. بالنسبة للتجربة التالية، سنحافظ على عدد الفترات محددًا بعشرين فترة للحصول على نتائج سريعة وأفضل.

```
if Model_Train:
history = model.fit(
x_train,
y_train,
validation_data=(x_val, y_val),
batch_size=64,
epochs=40,
verbose=2,
callbacks=[
tf.keras.callbacks.ReduceLROnPlateau(patience=5),
tf.keras.callbacks.ModelCheckpoint("Model/model.h5"),
],
)
Epoch 1/40
30/30 - 69s - loss: 0.4536 - val_loss: 0.3796
Epoch 2/40
30/30 - 57s - loss: 0.3856 - val_loss: 0.3601
Epoch 3/40
30/30 - 57s - loss: 0.3637 - val_loss: 0.3410
Epoch 4/40
30/30 - 57s - loss: 0.3488 - val_loss: 0.3255
Epoch 5/40
30/30 - 57s - loss: 0.3357 - val_loss: 0.3188
Epoch 6/40
30/30 - 57s - loss: 0.3295 - val_loss: 0.3163
```

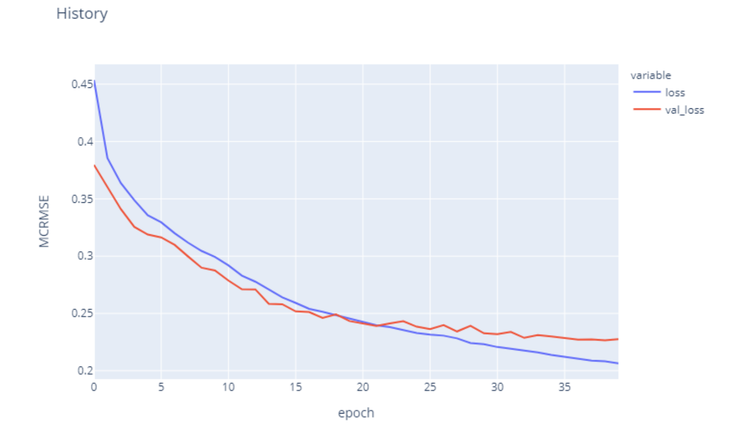
```
Epoch 7/40
30/30 - 57s - loss: 0.3200 - val_loss: 0.3098
Epoch 8/40
30/30 - 57s - loss: 0.3117 - val_loss: 0.2997
Epoch 9/40
30/30 - 57s - loss: 0.3046 - val_loss: 0.2899
Epoch 10/40
30/30 - 57s - loss: 0.2993 - val_loss: 0.2875
Epoch 11/40
30/30 - 57s - loss: 0.2919 - val_loss: 0.2786
Epoch 12/40
30/30 - 57s - loss: 0.2830 - val_loss: 0.2711
Epoch 13/40
30/30 - 57s - loss: 0.2777 - val_loss: 0.2710
Epoch 14/40
30/30 - 57s - loss: 0.2712 - val_loss: 0.2584
Epoch 15/40
30/30 - 57s - loss: 0.2640 - val_loss: 0.2580
Epoch 16/40
30/30 - 57s - loss: 0.2592 - val_loss: 0.2518
Epoch 17/40
30/30 - 57s - loss: 0.2540 - val_loss: 0.2512
Epoch 18/40
30/30 - 57s - loss: 0.2514 - val_loss: 0.2461
Epoch 19/40
30/30 - 57s - loss: 0.2485 - val_loss: 0.2492
Epoch 20/40
30/30 - 57s - loss: 0.2453 - val_loss: 0.2434
Epoch 21/40
30/30 - 57s - loss: 0.2424 - val_loss: 0.2411
Epoch 22/40
30/30 - 57s - loss: 0.2397 - val_loss: 0.2391
Epoch 23/40
30/30 - 57s - loss: 0.2380 - val_loss: 0.2412
Epoch 24/40
30/30 - 57s - loss: 0.2357 - val_loss: 0.2432
Epoch 25/40
30/30 - 57s - loss: 0.2330 - val_loss: 0.2384
Epoch 26/40
30/30 - 57s - loss: 0.2316 - val_loss: 0.2364
Epoch 27/40
30/30 - 57s - loss: 0.2306 - val_loss: 0.2397
Epoch 28/40
30/30 - 57s - loss: 0.2282 - val_loss: 0.2343
Epoch 29/40
30/30 - 57s - loss: 0.2242 - val_loss: 0.2392
Epoch 30/40
30/30 - 57s - loss: 0.2232 - val_loss: 0.2326
Epoch 31/40
30/30 - 57s - loss: 0.2207 - val_loss: 0.2318
Epoch 32/40
30/30 - 57s - loss: 0.2192 - val_loss: 0.2339
Epoch 33/40
30/30 - 57s - loss: 0.2175 - val_loss: 0.2287
Epoch 34/40
```

```
30/30 - 57s - loss: 0.2160 - val_loss: 0.2310
Epoch 35/40
30/30 - 57s - loss: 0.2137 - val_loss: 0.2299
Epoch 36/40
30/30 - 57s - loss: 0.2119 - val_loss: 0.2288
Epoch 37/40
30/30 - 57s - loss: 0.2101 - val_loss: 0.2271
Epoch 38/40
30/30 - 57s - loss: 0.2088 - val_loss: 0.2274
Epoch 39/40
30/30 - 57s - loss: 0.2082 - val_loss: 0.2265
Epoch 40/40
30/30 - 57s - loss: 0.2064 - val_loss: 0.2276
```

تقييم تاريخ التدريب

تم تقليل كل من خطأ التحقق من الصحة والتدريب حتى 20 فترة. أصبح خطأ التحقق من الصحة ثابتًا بعد 35، لذافي رأيي، يجب أن نختبر النتائجفي كل من 20 و35 فترة.

```
if Model_Train:
fig = px.line(
history.history,
y=["loss", "val_loss"],
labels={"index": "epoch", "value": "MCRMSE"},
title="History",
)
fig.show()
```



تحميل النماذج وعمل التنبؤات

تم تقسيم مجموعة بيانات الاختبار إلى مجموعات عامة وخاصة لها أطوال تسلسل مختلفة، لذلك من أجل التنبؤ بالتحلل على أطوال مختلفة، نحتاج إلى بناء نموذجين مختلفين وتحميل نقاط الحفظ checkpoints المحفوظة لدينا. هذا ممكن لأن نماذج RNN يمكن أن تقبل تسلسلات ذات أطوال متفاوتة كمدخلات.

سنقوم ببناء نموذجين مميزين بتسلسلات متباينة وأطوال توقع. يحتوي نموذجنا العام على 107 طول تسلسل بينما يحتوي نموذجنا الخاص على 130 طول تسلسل. سنقوم بتحميل وزننا المحفوظفي كلا النموذجين للتنبؤ بتحلل mRNA.

```
model_public = build_model(seq_len=107, pred_len=107,
embed_size=len(token2int))
model_private = build_model(seq_len=130, pred_len=130,
embed_size=len(token2int))
model_public.load_weights("Model/model.h5")
model_private.load_weights("Model/model.h5")
```

التنبؤ

لقد نجحنافي توقع كل من مجموعات البيانات العامة والخاصة.في الخطوة التالية، سنقوم بدمجها باستخدام معرف الاختبار test id.

```
public_preds = model_public.predict(public_inputs)
private_preds =
model_private.predict(private_inputs)private_preds.shape(3005, 130, 5)
```

المعالجة اللاحقة والإرسال

تحويل مصفوفة 3D NumPy إلى إطار بيانات:

- الجمع بين إطارات البيانات الخاصة والعامة.
- إضافة سلسلة من الأعداد الصحيحة أمام المعرف بناءً على سلسلة من التنبؤات الفردية على سبيل المثال [id_00073f8be_0، id_00073f8be_1، id_00073f8be_2 ..]
- دمج جميع البياناتفي إطار بيانات Pandasوالاستعداد للتسليم submission.

```
preds_ls = []
for df, preds in [(public_df, public_preds), (private_df,
private_preds)]:
for i, uid in enumerate(df.id):
single_pred = preds[i]
single_df = pd.DataFrame(single_pred, columns=target_cols)
single_df["id_seqpos"] = [f"{uid}_{x}" for x in
range(single_df.shape[0])]
preds_ls.append(single_df)
preds_df = pd.concat(preds_ls)
```

```
preds_df.head()
```

```
reactivity deg_Mg_pH10 deg_Mg_50C deg_pH10 deg_50C id_seqpos
0 0.685760 0.703746 0.585288 1.857178 0.808561 id_00073f8be_0
1 2.158555 3.243329 3.443042 4.394709 3.012130 id_00073f8be_1
2 1.432280 0.674404 0.672512 0.662341 0.718279 id_00073f8be_2
3 1.296234 1.306208 1.898748 1.324560 1.827133 id_00073f8be_3
4 0.851104 0.670810 0.971952 0.573919 0.962205 id_00073f8be_4
```

التسليم

دمج إطار بيانات العينة مع المتوقع على id_seqpos لتجنب التكرار والتأكد من أنه يتبع تنسيق التسليم. أخيرًا، احفظ إطار البيانات الخاص بنا في ملف csv..

```
submission = sample_df[["id_seqpos"]].merge(preds_df,
on=["id_seqpos"])
submission.to_csv("Submission/submission.csv",
index=False)submission.head()  id_seqpos reactivity deg_Mg_pH10
deg_Mg_50C deg_pH10 deg_50C
0 id_00073f8be_0 0.685760 0.703746 0.585288 1.857178 0.808561
1 id_00073f8be_1 2.158555 3.243329 3.443042 4.394709 3.012130
2 id_00073f8be_2 1.432280 0.674404 0.672512 0.662341 0.718279
```

Name	Submitted	Wait time	Execution time	Score
submission (10).csv	just now	1 seconds	3 seconds	0.27238

Complete

Jump to your position on the leaderboard

```
3 id_00073f8be_3 1.296234 1.306208 1.898748 1.324560 1.827133
4 id_00073f8be_4 0.851104 0.670810 0.971952 0.573919 0.962205
```

الاستنتاج

كانت هذه تجربة فريدة بالنسبة لي حيث كنت أتعامل مع ملفات JSON مع مصفوفات متعددةفي عينات واحدة. بعد معرفة كيفية استخدام البيانات، أصبح التحدي بسيطًا للغاية وأصبح لمجتمع Kaggle دور أكبرفي مساعدتيفي تحقيق ذلك. كانت هذه المقالة قائمة على النموذج تمامًا وبصرف النظر عن بناء النموذج، فقد استكشفت مجموعة البيانات واستخدمت تحليل البيانات لفهم بعض الأنماط الشائعة. كنت أرغبفي تضمين تجاربي مع نماذج أخرى لتعزيز التدرج وLSTM، ولكن بعد ذلك قررت تقديم أفضل نموذج ممكن.

لقد استخدمنا ملفات JSON وقمنا بتحويلها إلى مصفوفات Numpy ثلاثية الأبعاد رمزية ثم استخدمنا نموذج 3X GRU للتنبؤ بمعدل تدهور mRNA. أخيرًا، استخدمنا الأوزان المحفوظة لإنشاء نماذج مميزة لأطوال مختلفة من تسلسل RNA. سأقترح عليك استخدام الكود الخاص بي وتعديله للتحقق مما إذا كان بإمكانك التغلب على نتيجتيفي لوحة المتصدرين.

الكود:

https://dagshub.com/kingabzpro/mRNA-Vaccine-Degradation-Prediction

المصدر:

https://pub.towardsai.net/deep-learning-model-to-predict-mrna-degradation-1533a7f32ad4

19) تصنيف عائلة البروتين باستخدام نماذج التعلم العميقة PROTEIN FAMILY CLASSIFICATION USING THE DEEP LEARNING MODELS

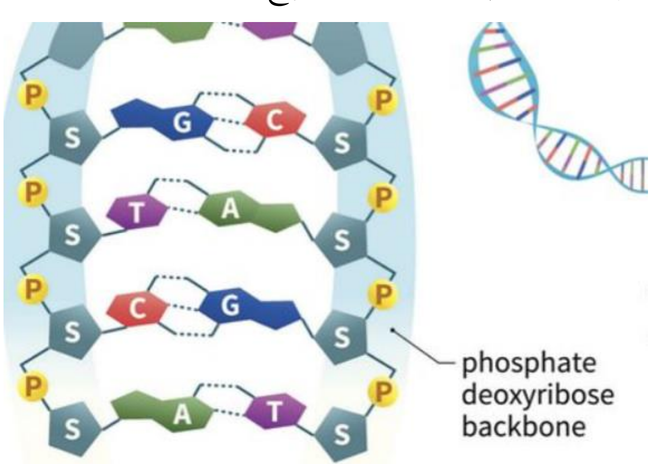
مقدمة:

يعد فهم العلاقة بين الأحماض الأمينية amino acid وتسلسل البروتين protein sequence مشكلة طويلة الأمدفي الاحياء molecular biology الجزيئية والآثار العلمية.

نقوم بتطوير نموذج التعلم العميق الذي يتعلم العلاقة بين تسلسلات الأحماض الأمينية غير المحاذاة unaligned amino acid sequences وتصنيفها الوظيفي عبر جميع قواعد بيانات PFAM البالغ عددها 17929 عائلة.

مشكلة العمل / العالم الحقيقي

المهمة هي: بالنظر إلى تسلسل الأحماض الأمينيةفي مجال البروتين، توقع الفئة class التي ينتمي إليها. يوجد حوالي مليون مثال تدريب، و17929 فئة إخراج.



نستخدم بيانات التسلسل غير المحاذاة unaligned sequence data لتدريب نماذج التعلم العميق لمعرفة التوزيع عبر عائلات البروتين للعثور على التحسين المشترك.

البيانات:

نقسم البيانات إلى مجموعة تدريب train وتطوير development ومجموعة اختبار test. نحن نستخدم بيانات التدريب لتدريب نماذجنا، ويمكن أن تكون بيانات التطوير للتحقق المتبادل cross validation، وبيانات الاختبار محفوظة للتقييمات.

لدينا مجالات مختلفةفي البيانات:

	family_id	sequence_name	family_accession	aligned_sequence	sequence
0	CPSase_L_D3	CARB_LACS1/422-540	PF02787.19	EKLFHAQDDRLFYIAEAF.RRG.Y.....TIE...........EV...	EKLFHAQDDRLFYIAEAF
1	SOCS_box	H2T4I2_TAKRU/225-262	PF07525.16	PPALMDLCA.......LAIQQ.HLGQQRHN............QI....	PPALMDLCALAIQQHLG(
2	Glug	Q8QNC2_ESV1K/681-707	PF07581.12	TNGRTGGVVGHAVG.........TDVTMCRNVA..TFT	TNGRTGGVVGHAVGTD'
3	PepSY	A8MEK6_ALKOO/172-230	PF03413.19	VISEEQA.KKIA...........LEK.............I....N....	VISEEQAKKIALEKINGK\
4	PMSR	D1AKJ7_SEBTE/3-153	PF01625.21	EIILAGGCFWGVEAYF.QRLN........G..VI...KTEVGYTDG...	EIILAGGCFWGVEAYFQI

وصف الميزات:

4. **sequence_name**: اسم التسلسل، بالصيغة "$ $ / uniprot_accession_id start_index- $ end_index".
5. **sequence**: عادةً ما تكون هذه هي ميزات الإدخالفي نموذجك. تسلسل الأحماض الأمينية لهذا المجال. هناك 20 نوعًا من الأحماض الأمينية الشائعة جداً (التردد> 1000000)، و 4 أحماض أمينية غير شائعة تمامًا: X، U، B، O، Z.
6. **Family_accession**: عادة ما تكون هذه هي التسميات الخاصة بنموذجك. رقم المدخل بالصيغة (Pfam) PFxxxxx.y ، حيث xxxxx هو اسم العائلة ، و y هو رقم الإصدار.
7. **family_id**: اسم العائلة من كلمة واحدة.
8. **aligned_sequence**: يحتوي على تسلسل واحد من محاذاة التسلسل المتعدد.

تحليل البيانات الاستكشافية

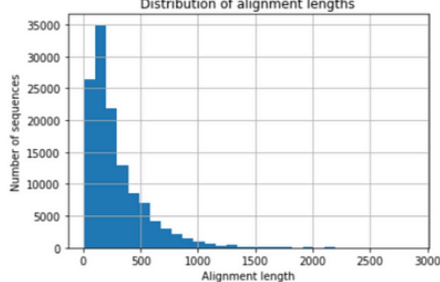
نقوم بتصور البيانات لفهم البيانات ولإزالة القيم المتطرفة outliers الموجودةفي البيانات. نقوم أيضًا بتحليل سلوك البيانات حتى نتمكن من القيام باستخراج الميزات feature extraction وهندسة الميزات feature engineering فوق البيانات.



مخطط الكمان لطول التسلسل.

```
In [7]:
dev['alignment_length'] = dev.aligned_sequence.str.len()
dev.alignment_length.hist(bins=30)
plt.title('Distribution of alignment lengths')
plt.xlabel('Alignment length')
plt.ylabel('Number of sequences')

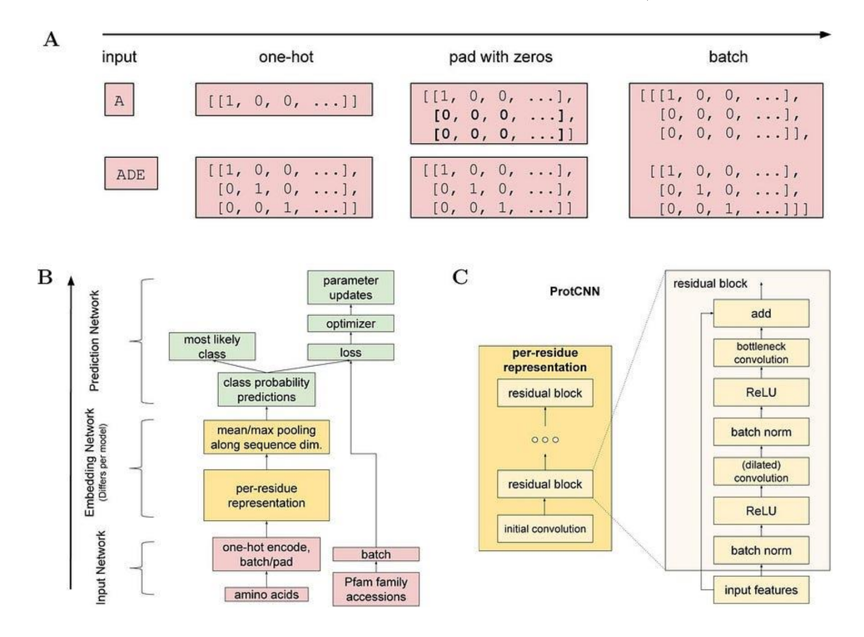
Out[7]:
Text(0, 0.5, 'Number of sequences')
```



توزيع الأطوال المحاذاة.

استخراج الميزات وهندسة الميزات:

تقوم شبكة الإدخال بتعيين تسلسل من الأحماض الأمينية L إلى مجموعة ثنائية (L,26) والتي ليست سوى الأبجدية الموجودةفي الأحماض الأمينية، حيث يمثل كل عمود تمثيل الأحماض الأمينية.



يتم تبطين التسلسلات بطول أطول تسلسل في الدُفعة إلى مصفوفة (L,26) تحتوي على تمثيل واحد من الأحماض الأمينية ذات الترميز واحد ساخن لمصفوفة التسلسل (L,F) التي تحتوي على التضمين لبقايا التسلسل. تم تصميم جميع عمليات المعالجةفي شبكة التضمين اللاحقة بحيث تكون ثابتة على الحشو الذي تم تقديمه لتسلسل معين.

```
import string
def string_vectorizer(strng, alphabet=string.ascii_lowercase):


    vector = [[0 if char != letter else 1 for char in alphabet]
                  for letter in strng]
    vector1=np.array(vector)
    shapeout=vector1.shape[0]
    diff=600-shapeout
    reshapearray=np.zeros((diff,26),dtype=int)
    lenarray=len(strng)
    finalarray=np.vstack((vector,reshapearray,newarray))
    return finalarray
trainarray1=[]
for sen in traindataframe['sequence']:
   trainarray1.append(string_vectorizer(sen.lower()))


testarray1=[]
for sen in testdataframe['sequence']:
    testarray1.append(string_vectorizer(sen.lower()))


cvarray1=[]
for sen in cvdataframe['sequence']:
    cvarray1.append(string_vectorizer(sen.lower()))


trainarray1=np.array(trainarray1).reshape(len_of_traindata,600,26)
testarray1=np.array(testarray1).reshape(len_of_testdata,600,26)
```

```
cvarray1=np.array(cvarray1).reshape(len_of_cvdata,600,26)
```

لدينا تسميات الفئات التي ليست سوى الوصول إلى family_accession في البيانات هي تسمية مشفرة لتعيينها كمتغيرات الهدف target variables.

```
from sklearn.preprocessing import LabelEncoder
label1=LabelEncoder()
label1.fit(totaldataframe['family_accession'])
trainy=label1.transform(traindataframe['family_accession'])
testy=label1.transform(testdataframe['family_accession'])
cvy=label1.transform(cvdataframe['family_accession'])
```

تدريب النموذج

تستخدم شبكات ProtCNN الخاصة بنا الشبكات المتبقية (ResNets)، وهو نوع من الشبكات العصبية التلافيفية CNN التي تتدرب بشكل أسرع وأكثر استقرارًا)، حتى مع وجود العديد من الطبقات، تعتبر شبكات ProtCNN ثابتة من حيث الترجمة، وهي ميزة للعمل مع بيانات تسلسل البروتين غير المحاذاة. العديد من المخلفات. يأخذ الالتفاف 1d المتوسع n عمليات التفاف قياسية على كل عنصرفي تسلسل، مما يسمح بجمع المعلومات المحلية والعالمية دون زيادة كبيرةفي عدد معلمات النموذج.

```
import keras as keras
import tensorflow as tf
from keras.models import Sequential
from keras.layers import Dense,Dropout,Flatten,BatchNormalization
from keras.layers import Conv1D,MaxPooling1D
model=Sequential()
model.add(Conv1D(2000,kernel_size=26,activation='relu',input_shape=(20
0,26)))
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(0.4))
model.add(BatchNormalization())
model.add(Conv1D(250,20,activation='relu'))
model.add(MaxPooling1D(pool_size=4))
model.add(Dropout(0.6))
```

```
model.add(BatchNormalization())

model.add(Flatten())

model.add(Dense(11836,activation='softmax'))

model.compile(loss=keras.losses.sparse_categorical_crossentropy,optimi
zer='adam',metrics=['accuracy'])

history=model.fit(trainarray,trainy,epochs=30,validation_data=[testarr
ay,testy],batch_size=128)
```

نستخدم مُحسِّن آدم Adam optimizer، حيث يخضع معدل التعلم للاضمحلال الأسي exponential decay، وفي وقت التدريب نقدم النموذج بدُفعات مرسومة عشوائيًا.

من أهم المعامِلات الفائقة hyperparameter المركّبة هو حجم المجال الاستقبالي receptive field size لكل ميزة لكل بقايا، والتي تصف طول التكرارات اللاحقة التي تؤثر على قيمتها. يتيح استخدام التلافيف المتوسعة الحصول على أحجام مجال استقبالية أكبر دون حدوث انفجارفي عدد معلمات النموذج. على حد علمنا، هذا هو أول تطبيق للتلافيف المتوسعة لتصنيف تسلسل البروتين. ثم يتم تجميع المصفوفة (L,F) على طول التسلسل لضمان الثبات في الحشو.

BiLSTM هو امتداد لـ LSTM، حيث يبدأ التكرار الإضافي من الخطوة الزمنية الأخيرة للتكرار الأمامي وينتقل للخلف إلى الخطوة الزمنية الأولى للتكرار الأمامي. وبالتالي يمكن التقاط المعلومات الواردة في الخطوات "المستقبلية" وتساعد في عمل التنبؤات في خطوات زمنية سابقة.

```
import keras as keras

import tensorflow as tf

from keras.models import Sequential

from keras.layers import Dense,Dropout,Flatten,BatchNormalization

from keras.layers import Conv1D,MaxPooling1D

from keras.layers import LSTM,Bidirectional

model=Sequential()

model.add(Dense(200,activation='relu',input_shape=(200,26)))

model.add(Dropout(0.4))

model.add(BatchNormalization())

model.add(Bidirectional(LSTM(26,dropout=0.2,return_sequences=True)))

model.add(BatchNormalization())

model.add(Flatten())
```

```
model.add(Dense(11836,activation='softmax'))

model.compile(loss=keras.losses.sparse_categorical_crossentropy,optimizer='adam',metrics=['accuracy'])

history=model.fit(trainarray,trainy,epochs=3,validation_data=[testarray,testy],batch_size=128)
```

الاستنتاج

يمكن أن تؤدي زيادة عدد معلمات النموذج عبر عدد الفلاتر وحجم النواة وزيادة حجم الدُفعة إلى تحسيناتفي الأداء. بشكل أساسي، كانت بصمة ذاكرة النماذج التي دربناها محدودة بمقدار الذاكرة المتاحة على وحدة معالجة رسومات واحدة، مما استلزم إجراء مقايضات بين هذه العوامل المختلفة. من بين التجارب التي أجريناها أفضل أداء لـ ProtCNN لـ Pfam الكامل يتكون من كتلة متبقية واحدة مع 2000 فلتر، وحجم نواة 26، وحجم دفعة 128.

بالإضافة إلى نماذج CNN، قمنا أيضًا بتدريب شبكة عصبية متكررة (RNN) ذات طبقة واحدة ثنائية الاتجاه LSTM، والتي حققت دقة قدرها 0.982 على مجموعة بيانات Pfam.

المصدر:

https://vignesh943628.medium.com/protein-family-classification-using-the-deep-learning-models-38f4acd35f14

20) تصنيف تسلسل البروتين متعدد الفئات باستخدام التعلم العميق
Protein sequence multi-class classification using deep learning

مقدمة

يعد فهم العلاقة بين تسلسل الأحماض الأمينية amino acid sequence ووظيفة البروتين protein function مشكلة طويلة الأمدفي الاحياء الجزيئية molecular biology ذات آثار علمية بعيدة المدى. على الرغم من ستة عقود من التقدم، لا يمكن للتقنيات الحديثة أن تشرح ثلث تسلسل البروتينات الجرثومية، مما يعيق قدرتنا على استغلال التسلسلات التي تم جمعها من كائنات متنوعة. لمعالجة هذا، أبلغنا عن نموذج التعلم العميق deep learning model الذي يتعلم العلاقة بين تسلسل الأحماض الأمينية غير المحاذاة وتصنيفها الوظيفي عبر جميع 17929 عائلة من قاعدة بيانات Pfam. يقوم نموذجنا بتحديد موقع التسلسلات من العائلات غير المرئيةفي مساحة التضمين، مما يسمح بتوضيح التسلسلات من العائلات الجديدة بدقة. تشير هذه النتائج إلى أن نماذج التعلم العميق ستكون مكونًا أساسيًافي أدوات التنبؤ بوظيفة البروتينفي المستقبل. يعد توقع وظيفة البروتين من تسلسل الأحماض الأمينية الخام خطوة حاسمة لفهم العلاقة بين النمط الجيني والنمط الظاهري. مع انخفاض تكلفة تسلسل الحمض النووي وازدهار مشاريع التسلسل الميتاجينومي metagenomic sequencing، ستلعب الأدوات السريعة والفعالة التي تعلق على إطارات القراءة المفتوحة مع الوظيفة دورًا رئيسيًافي استغلال هذه البيانات.

1. مشكلة العمل/ العالم الحقيقي

تصنيف تسلسل البروتين الخاص بالحمض الأميني إلى أحد فصائل العائلة family accession.

يمكن استخدام هذا النموذج للتنبؤ بتسلسل بروتين معين من الأحماض الأمينية. سيولد النموذج عددًا من قيم احتمالية الفئات المقابلة لعدد الفئة أو فصائل العائلة. ستكون أعلى قيمة احتمالية للفئة المقابلة هي الفئة المتوقعة لتسلسل البروتين من الأحماض الأمينية.

2. الأهداف والقيود

الاهداف:

هدفنا هو توقع تسلسل البروتين المحدد للحمض الأميني بأكبر قدر ممكن من الدقة.

القيود:

1. **التفسير Interpretability:** التفسير مهم لتسلسل البروتين من الأحماض الأمينية التي يجب أن تتنبأ بها بشكل صحيح.
2. **التأخير Latency:** بالنظر إلى تسلسل البروتين للحمض الأميني، يجب أن يتنبأ بالفئة الصحيحة لذلك ليست هناك حاجة إلى زمن تأخير عالٍ.

3. **الدقة Accuracy:** هدفنا هو توقع تسلسل البروتين المحدد للحمض الأميني بأكبر قدر ممكن من الدقة. كلما زادت دقة الاختبار، كان أداء نموذجنا أفضل في العالم الحقيقي.

3. مقاييس الأداء

هذه مشكلة تصنيف متعددة الفئات multi-class classification مع 17929 فئة مختلفة، لذلك فقد أخذنافي الاعتبار مقياسين للأداء:

1. **خطأ السجل متعدد الفئات Multi-Class Log-loss:** لقد استخدمنا نموذج التعلم العميق مع طبقة الانتروبيا المتقاطعة cross-entropy layerفي النهاية مع 17929 وحدة softmax، لذلك، هدفنا هو تقليل خطأ السجل متعدد الفئات / خطأ الانتروبيا المتقاطعة.
2. **الدقة Accuracy:** يخبرنا هذا بمدى دقة أداء نموذجنافي توقع التعبيرات expressions.

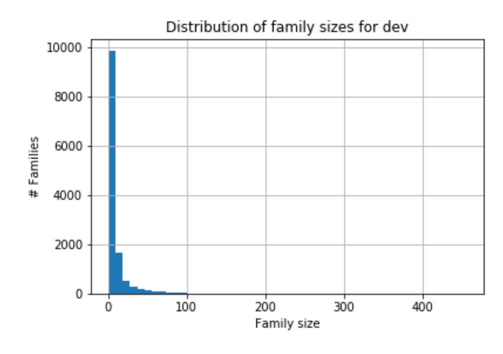
4. بيانات المصدر

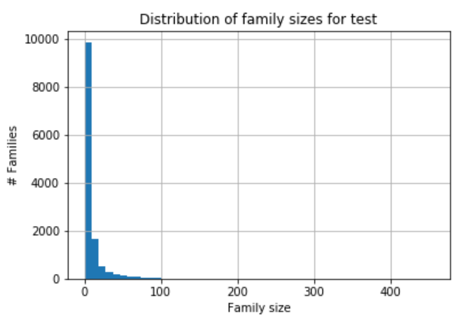
يتم توفير البيانات بواسطة kaggle حتى نتمكن من التنزيل مباشرة من رابط Kaggle المحدد ـ https://www.kaggle.com/googleai/pfam-seed-random-split

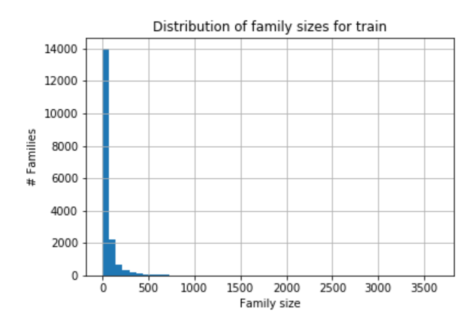
5. نظرة عامة على البيانات

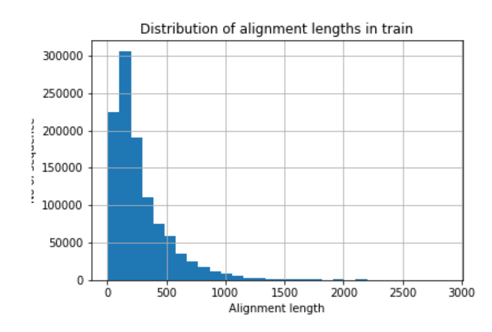
الطريقة المستخدمة لتقسيم البيانات إلى طيات folds تدريب training / تطوير dev/ اختبار testing هي تقسيم عشوائي.

- يجب استخدام بيانات التدريب لتدريب نماذجك.
- يجب استخدام بيانات التطوير Dev (التطوير development)في حلقة تحقق قريبة (ربما لضبط المعلمة الفائقة hyperparameter tuning أو التحقق من صحة النموذج model validation).
- يجب حجز بيانات الاختبار للتقييمات الأقل تكرارًا ـ وهذا يساعدفي تجنب الضبط الزائد overfitting على بيانات الاختبار الخاصة بك، حيث يجب استخدامها بشكل غير متكرر.









توزيع أطوال المحاذاةفي التدريب

محتوى الملف

كل طية (تدريب، مطور، اختبار) بها عدد من الملفات. يحتوي كل ملف من هذه الملفات على ملف CSVفي كل سطر، والذي يحتوي على الحقول التالية:

sequence: *HWLQMRDSMNTYNNMVNRCFATCIRSFQEKKVNAEEMDCTKRCVTKFVGYSQRVALRF AE*

family_accession: *PF02953.15*

sequence_name: *C5K6N5_PERM5/28–87*

aligned_sequence: *....HWLQMRDSMNTYNNMVNRCFATCI...........RS.F....QEKKVNAEE.....MDCT....KRCVTKF VGYSQRVALRFAE*

family_id: *zf-Tim10_DDP*

وصف الحقول:

- **sequence**: عادةً ما تكون هذه هي ميزات الإدخالفي نموذجك. تسلسل الأحماض الأمينية لهذا المجال. هناك 20 نوعًا من الأحماض الأمينية الشائعة جداً (التكرار frequency > 1000000)، و 4 أحماض أمينية غير شائعة تماماً: X، U، B، O، Z.

- **Family_accession**: عادة ما تكون هذه هي التسميات الخاصة بنموذجك. رقم المدخل بالصيغة (Pfam) PFxxxxx.y ، حيث xxxxx هو فئة العائلة ، و y هو رقم الإصدار. بعض قيم y أكبر من عشرة، وبالتالي فإن "y" تتكون من رقمين.
- **family_id**: اسم من كلمة واحدة للعائلة.
- **sequence_name**: اسم التسلسل ، بالصيغة:
  "$uniprot_accession_id/$start_index-$end_index"
- **align_sequence**: يحتوي على تسلسل واحد من محاذاة تسلسل متعدد (مع بقية أفراد الأسرةفي البذرة، مع الاحتفاظ بالفجوات).

بشكل عام، يعتبر `family_accession` هو التسمية label، والتسلسل `sequence` (أو التسلسل المحاذي aligned sequence) هو الميزة المدربة.

هذا التسلسل يتوافق مع مجال، وليس بروتين كامل.

محتويات هذه الحقول هي نفسها البيانات المقدمةفي تنسيق ستوكهولم بواسطة PFam

atftp: //ftp.ebi.ac.uk/pub/databases/Pfam/releases/Pfam32.0/Pfam-A.seed.gz

6. المكتبات والحزم

لقد استخدمنا تقريبًا جميع مكتبات التعلم العميق التي استخدمناها عادةً مثل pandas، و numpy، و sklearn ، و Keras ، وما إلى ذلك. ويمكننا تثبيتها ببساطة باستخدام

```
pip install 'هنا الحزمة اسم تحديد'
```

7. المعالجة المسبقة والتخصيص

لقد أجرينا بعض المعالجة المسبقة مثل التسلسل بالأحرف الصغيرة واستخدمنا الحد الأقصى لطول التسلسل البالغ 100 وتحويل البيانات النصية إلى شكل رقمي باستخدام sklearn countvectorizer أو دالتنا المحددة لإنشاء قاموس وتعيين الحرف إلى نموذج الفهرس والفهرس إلى شكل حرف.

8. النمذجة والتدريب

لذا فقد وصلنا أخيرًا إلى آخر عملية لملائمة النموذج model fit. وبذلك نكون قد أكملنا 90٪ من العملية. لذا فإن هدفنا الرئيسي هنا هو إعطاء تسلسل البروتين واحداً تلو الآخر وتقليل خطأ الانتروبيا. لقد قمنا بتصميم بُنية الشبكة العصبية الخاصة بنا على النحو التالي:

```
input_s = Input(shape=(100,24))
X = Conv1D(32, 1 , strides=1,padding='valid', name='conv1d_1',
kernel_initializer=glorot_uniform(seed=0))(input_s)
X = MaxPooling1D(pool_size=2)(X)
```

```
X1 = BatchNormalization(axis=2, name='batch_normalization_1')(X)
X2 = Activation('relu',name='activation_1')(X1)
X3 = BatchNormalization(axis=2, name='batch_normalization_2')(X2)
X4 = Activation('relu',name='activation_2')(X3)
X5 = Conv1D(128, 1 , strides=1,padding='valid', name='conv1d_2',
kernel_initializer=glorot_uniform(seed=0))(X4)
X6 = BatchNormalization(axis=2, name='batch_normalization_3')(X5)
X7 = Activation('relu',name='activation_3')(X6)
X8 = Conv1D(128 , 1 , strides=1 ,padding='valid', name='conv1d_3' ,
kernel_initializer=glorot_uniform(seed=0))(X7)
X8 = Dropout(0.5,name='d1')(X8)
X8 = MaxPooling1D(pool_size=2)(X8)
X9 = Conv1D(128, 1 , strides=1 ,padding ='valid',name='conv1d_4',
kernel_initializer=glorot_uniform(seed=0))(X2)
X9 = Dropout(0.5,name='d2')(X9)
X9 = MaxPooling1D(pool_size=2)(X9)
X10 = Add()([X8,X9])
X11 = Activation('relu',name='activation_4')(X10)
X11 = Dropout(0.2,name='d3')(X11)
X12 = BatchNormalization(axis=2,name='batch_normalization_4')(X11)
X13 = Activation('relu',name='activation_5')(X12)
X14 = Dropout(0.5,name='d4')(X13)
X15 = Flatten(name='flatten_1')(X14)
X16 = Dense(2910 ,name='fc' + str(2910), kernel_initializer =
glorot_uniform(seed=0))(X15)
X17 = Activation('softmax',name='activation_6')(X16)
```

دعونا نفهم المعمارية:

كما نرى، لدينا شكل إدخال (100,24)أي (None,100,24)نظرًا لأن لدينا حدًا أقصى لطول التسلسل 100 لكل تسلسل بروتين من الأحماض الأمينية وهو واحد مشفر ساخن one hot encoded، لذا فهو على سبيل المثال:

```
for eg: we have shape (1,2,3)
so one hot encode of this look like
[[1 0 0],[0 1 0],[0 0 1]]
```

هنا 1 وصف مكان تقديم فهرس char. وبهذه الطريقة، يتم تشفير وتمثيل جميع بيانات التدريب والاختبار والتطوير الخاصة بنا.

من المعمارية، يمكننا أن نلاحظ أن لدينا 4 طبقات التفاف convolution layer أحادية البعد 1D، وطبقة كثيفة Dense layer، وطبقة التجميع Maxpooling، وطبقة التسرب Dropout، وطبقة BatchNormalization، وطبقة التنشيط Activation، وطبقة التسوية Flatten. لقد حددنا تهيئة الوزن weight initialization. إذا لم تحدد طريقة تهيئة الوزن بشكل صريح في Keras، فإنها تستخدم

تهيئة Xavier المعروفة أيضًا باسم تهيئة Glorot. الهدف من كل هذه العوامل المبدئية للوزن هو إيجاد تباين جيد للتوزيع الذي يتم من خلاله استخلاص المعلمات الأولية.

حول طبقة الالتفاف،

```
X  = Conv1D(32, 1 , strides=1,padding='valid', name='conv1d_1',
kernel_initializer=glorot_uniform(seed=0))(input_s)
```

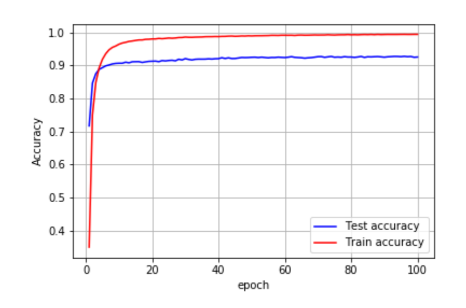
هنا 32 حدد 32 فلتراً بحجم نواة 1 نظرًا لأننا استخدمنا فلتراً مختلفًافي طبقة الالتفاف حتى تعلمت الشبكة معلمة أفضل. لقد استخدمنا "padding = "valid لذلك لن يكون هناك حشوة.

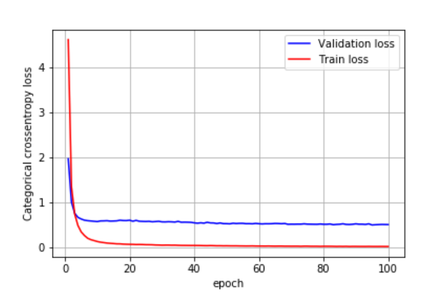
لقد استخدمنا طبقة التسرب بين طبقة الالتفاف لأنها تساعد على منع الضبط الزائد overfitting. لقد استخدمنا هنا معدل تسرب dropout rate مختلف، أي 0.5 حددنا أنه سيتم حذف 0.5 ٪ من العقد أثناء التمرير الأمامي والخلفي نظرًا لعدم معرفة أي معلمة أثناء الانتشار الخلفي backpropagation. نظرًا لأن الشبكة تزيل الوحدات مؤقتًا من التوصيلات، فإن إزالة عدد العقد للاتصال يكون عشوائيًا. حتى نتمكن من الحصول على معدل التسرب الأمثل عن طريق الضبط الفائق hyper-tuning أو استخدام معدلات التسرب المختلفة.في حالتنا، لاحظت خلال الفترات أننا كنا نعاني من الضبط الزائد لذا حاولت بمعدل تسرب صغير إلى كبير لتجنب الضبط الزائد (0.1ـ0.5).

بعد ذلك، لدينا طبقة max-pooling حيث يتم تقليل الميزات بطريقة هادفة تساعدفي اختزال العينة downsample من طبقة الالتفاف وكما نعلم مزايا مثل ثبات الموقع location invariance وثبات النطاق scale invariance لطبقة max-pool.في شبكتنا، استخدمنا طبقة تجميع أحادية البعد 1D max pool بحجم 2 أثناء التكرار عبر خريطة المعالم feature map. لفهم المزيد عن الشبكة العصبية التلافيفية والطبقات، قم بزيارة هذا الرابط.

أخيرًا، لدينا طبقة كثيفة تحتوي على وحدات تنشيط relu. نظرًا لأن طبقة الإخراج تحتوي على 2910 وحدة softmax. سيولد احتمالية فئة 2910 والتي تكون مجموعها 1. سنقوم بتقليل الخطأ عن طريق إعادتها أثناء الانتشار الخلفي. وبهذه الطريقة، سيتم تدريب نموذج شبكتنا العصبية على تصنيف تسلسل البروتين.

التدريب لمدة 100 فترة epochs دون ضبط الشبكة حصلت على النتائج التالية:





مخطط الدقة والخطأ

كما يمكننا أن نلاحظ أن الخطأ والدقة لهما فرق بين بيانات التدريب والاختبار، أي الضبط الزائد overfitting.

بعد ضبط التسرب dropout وإضافة طبقة التجميع القصوى max-pooling layer وتسوية الدُفعات batch normalization، حصلت على دقة اختبار تبلغ ٪99.

إذن، كيف حققت دقة تزيد عن ٪90 أقل من نموذج مختلف مع ضبط المعلمة الفائقة hyperparameter tuning:

```
+--------------------------------------------------+----------+
|                      Model                       | Accuracy |
+--------------------------------------------------+----------+
|          CNN without hyperparameter tuning       |  0.8423  |
|        CNN with dropout and BatchNomalization    |  0.9211  |
| CNN with dropout,BatchNormalization & Max-pooling |  0.9434  |
|         Final CNN with hyperparameter tuning     |  0.9922  |
+--------------------------------------------------+----------+
```

9. نتائج الاختبار

نظرًا لأن لدينا بيانات بتنسيق train، test، dev. لذلك استخدمنا بيانات الاختبار للتحقق من الدقة.

بعد الاختبار حصلنا على هذه النتائج النهائية:

```
score = model.evaluate(finaltest,ytest , verbose=1)
print("Test loss:",score[0])
print("Test accuracy:",score[1])#output
7316/7316 [==============================] - 2s 251us/step
Test loss: 0.0511749743080253
Test accuracy: 0.9922886276653909
```

10. اختبار في العالم الحقيقي

للاختبار، يجب اتباع نفس العملية:

1. المعالجة المسبقة للتسلسل بأحرف صغيرة وأخذ أقصى طول للتسلسل يبلغ 100.
2. تعيين حرف إلى فهرس أو شكل رقمي.
3. استخدام نموذجنا للتنبؤ النهائي بتسلسل بروتين من الأحماض الأمينية.

```
def predict(text):
  text=text.lower()
  final=[]
  seq1=[]
  for s in (text):
    x=train_char_index_dict[s]
    seq1.append(str(x))
  final.append(seq1)
  final_sequence =sequence.pad_sequences(final,
                                        maxlen=100,padding='post')
  nb_classes = 24
  targets = np.array(final_sequence)
  one_hot_train = np.eye(nb_classes)[targets]
  res=model.predict(one_hot_train)
  pred = labelencoder.inverse_transform([np.argmax(res)])
  return ("Given protein sequence '{}' belongs to class family
                       accession
{}".format(text.upper(),pred[0]))predict('hcqltgrqpgfghhishshrrtkrr
fdpniqhkrywlpsegrhirltlstkaiktvdti')#output
"Given protein sequence
'HCQLTGRQPGFGHHISHSHRRTKRRFDPNIQHKRYWLPSEGRHIRLTLSTKAIKTVDTI'
belongs to class family accession PF0083019"
```

11. مزيد من النطاق

نظرًا لأننا حصلنا على نتيجة جيدة ولكن يمكن تحسينها بشكل أكبر:

1. من أجل الحصول على مزيد من الدقة يمكن تدريب النموذج على مجموعة بيانات كبيرة.
2. من خلال المزيد من ضبط المعلمات الفائقة، يمكن تحقيق المزيد من الدقة.
3. يمكن استخدامه لمشكلة تصنيف تسلسل البروتين الأخرى عن طريق ضبط أو زيادة تعقيد النموذج.

المصدر:

https://medium.com/@jangidajay271/pfam-seed-random-split-protein-sequence-multi-class-classification-using-deep-learning-a0b172015c67

Deep Learning

In

Bioinformatics

20 Deep Learning Projects in Bioinformatics Solved and Explained with Python

By: Dr. Alaa Taima

